Diplomarbeit

zur Erlangung des akademischen Grades

Diplom-Informatiker (FH)

an der
Fachhochschule Konstanz
– Hochschule für Technik, Wirtschaft und Gestaltung –
im Fachbereich Informatik/Wirtschaftsinformatik

Thema:	Entwicklung eines Einsatzabwicklungssystems für Feuerwehren EASy
Diplomand:	**Stefan Michael Tröndle** Am Posthalterswäldle 54 78224 Singen
Betreuer:	Prof. Dr. Paul Wenzel FH Konstanz
ext. Betreuer:	Dipl.-Ing. (FH) Andreas Egger Freiwillige Feuerwehr Singen Hauptstraße 31 78224 Singen
Eingereicht:	Konstanz, den 18. September 2003

Abstract

Kurzzusammenfassung des Themas:
Nachdem bei vielen kleineren und mittelgroßen Feuerwehren das komplette Berichtswesen für Einsätze immer noch auf Papier geführt wird und Statistik oft eine Sache von wochenlanger Handarbeit ist, wurde im Rahmen dieser Diplomarbeit ein System entwickelt, das auf Basis einer Client-Server-Architektur die vernetzte Echtzeit-Bearbeitung von Feuerwehr-Einsätzen erlaubt. Dabei wurde Wert auf offene Schnittstellen zu Fremdsystemen gelegt und einige dieser Schnittstellen wurden auch implementiert.
Die Diplomarbeit ist gegliedert in eine allgemeine und spezielle Einführung (Kapitel 1-3) und eine Abhandlung über die Entwicklung der Software und deren Architektur und Funktionalität (Kapitel 4). Eine mögliche Vermarktungsstrategie für das System wurde ebenfalls konzipiert (Kapitel 5).

Schlagworte:
Feuerwehr
Dokumentation / Einsatzprotokoll
.NET-Plattform, C#
Messaging, Microsoft Message Queueing
Datenbank, MySQL
Linux, MONO-Projekt

Widmung

Diese Diplomarbeit ist allen Menschen in Uniform gewidmet, die jeden Tag bei der Arbeit viel mehr riskieren als nur ihre Karriere.
Wir sind Ihnen allen verpflichtet.

Inhaltsverzeichnis

Abbildungsverzeichnis

Quellcode-Verzeichnis

1 Einleitung

1.1 Die Freiwillige Feuerwehr Singen

Die Freiwillige Feuerwehr Singen wurde 1866 gegründet und besteht aus insgesamt sieben Abteilungen (Kernstadt, Beuren an der Aach, Bohlingen, Friedingen, Hausen an der Aach, Schlatt unter Krähen und Überlingen am Ried). In diesen sieben Abteilungen verrichten im Moment ca. 270 Bürgerinnen und Bürger freiwillig Ihren Dienst. Insgesamt 3 hauptamtliche Kräfte sind für Wehrführung, Verwaltung und Gerätewartung verantwortlich. Neben den aktiven Einsatzkräften gibt es noch Altersabteilungen und Jugendfeuerwehrabteilungen. Im Jahr sind im gesamten Stadtgebiet ca. 300 Einsätze abzuwickeln, wobei ca. 95% davon auf den Bereich Kernstadt entfallen.

1.2 Über mich und meine Qualifikation

Ich selbst bin seit 1991 Mitglied der Freiwilligen Feuerwehr Singen, zuerst in der Jugendfeuerwehr, seit 1997 Aktives Mitglied in der Abteilung Kernstadt. Ich bin in diversen Sondereinheiten (Gefahrgut, Führungsgruppe) tätig und kenne daher den organisatorischen Aufwand, der hinter der Bewältigung eines Einsatzes steckt, sehr gut.
Sicherlich beziehe ich einen Großteil meines Wissens über die Abläufe aus meiner persönlichen Erfahrung mit und bei der Freiwilligen Feuerwehr Singen. Durch meine langjährige Tätigkeit im Katastrophenschutzstab des Landkreises Konstanz und für den Kreisfeuerwehrverband Konstanz e.V. ist es mir jedoch gelungen, weit über den Tellerrand hinaus zu schauen und Erfahrung zu sammeln, wie in anderen Feuerwehren Gefahrenabwehr betrieben wird.

1.3 Motivation

Durch die oben erwähnte Mitarbeit in diversen Sondereinheiten entstand bei mir immer mehr das Bedürfnis, die vorhandene Organisation zu optimieren und zu unterstützen, um den Bürgern eine noch bessere Qualität der Betreuung durch

die Feuerwehr zu bieten. Dass eine Software hier nur der Tropfen auf den heissen Stein ist, ist mir bewusst, jedoch ist es mir in meinen Augen gelungen, ein Hilfsmittel zu schaffen, das die menschlichen Entscheidungsträger von Routineaufgaben entlastet.

1.4 Dankesworte

Mein Dank gilt allen, die mich bei der Erstellung dieser Diplomarbeit tatkräftig unterstützt haben, sei es durch Steigerung meiner Motivation, durch Anregungen oder konkrete Tätigkeiten.
Besonders bedanken möchte ich mich bei meinen Eltern Roland und Marianne Tröndle für die Unterstützung, bei Andreas Martin für seinen unermüdlichen Einsatz beim Korrekturlesen und seine wichtigen Ratschläge (auch wenn ich sie nicht immer angenommen habe), bei Hendrik Roggendorf für sein stetiges Drängen auf Vollendung dieser Diplomarbeit, bei Jens Jurkschaft für die Versorgung mit aktuellen Informationen rund um die Funktechnologie und natürlich bei Andreas Egger, der mich hervorragend betreut hat.
Danke, ohne Euch hätte ich es nicht in dieser Form geschafft.

Besonderer Dank gilt auch meinem betreuenden Professor, Dr. Paul Wenzel, der ohne zu zögern dieses nicht alltägliche Thema zur Betreuung angenommen hat.

Singen, im September 2003

Stefan Tröndle

Teil I
Hauptteil

2 Einführung

2.1 Was ist eigentlich "EASy" ?

EASy ist ein **E**insatz-**A**bwicklungs-**Sy**stem für Feuerwehren. Dieses Akronym wurde gewählt, einerseits, um den Benutzern die Berührungsängste vor dem System zu nehmen, andererseits auch um eine einprägsame Marke zu schaffen. EASy ist ein einsatzunterstützendes System für Feuerwehren, die keine eigene Notrufabfrage[1] durchführen. Gerade in diesem Bereich ist selten eine Softwareunterstützung des jeweiligen Zentralisten (in der jeweiligen Feuerwehr) gegeben. Sinn macht EASy wenn ein Einsatzaufkommen von mindestens 100 Einsätzen pro Jahr vorliegt, da nur dann der aufzuwendende Schulungs-, Wartungs- und Pflegeaufwand in Relation zur entstehenden Zeitersparnis steht. In den entsprechend oben genannten Feuerwehren übernehmen oft viele verschiedene Einsatzkräfte die Zentralistenfunktion. Diese Einsatzkräfte sind in der Regel nicht speziell für diese Funktion ausgebildet, daher kann hier ein komplexes System mit hohem Schulungsaufwand nur begrenzt zum Einsatz kommen. Einerseits beschränkt diese Forderung den möglichen Leistungsumfang des Systems. Andererseits stellte es mich als Entwickler und potentiellen Anwender auch vor die Aufgabe, ein System zu entwickeln, das nicht nur intuitiv zu bedienen, sondern trotz allem die benötigten Funktionalitäten bietet und genügend Spielraum für einen weiteren Ausbau lässt.

2.2 Was leistet das System ?

Das System soll den Zentralisten in der Funkzentrale von Routineaufgaben entlasten und ihm[2] bei der Durchführung seiner Tätigkeit zur Seite stehen[3]. Es ist *nicht* als Einsatzleitsystem[4] zu verstehen, dies ist auch nicht beabsichtigt. Es soll die Protokollierung des Einsatzgeschehens erleichtern, Telefonlisten und Objektpläne zur Verfügung stellen und helfen, bei mehreren gleichzeitig ablau-

[1] Notrufabfrage: Anrufe auf der Nummer "112" werden hier angenommen und weiterverarbeitet

[2] auf die explizite Nennung der weiblichen Form wird in dieser Diplomarbeit konsequent verzichtet, was keine Wertung darstellt, sondern lediglich der besseren Lesbarkeit dient.

[3] Was diese Tätigkeit beinhaltet, ist dem Kapitel *Situationsbeschreibung* zu entnehmen

[4] Mehr zur Definition siehe unten

fenden Einsätzen, den Überblick zu bewahren, also operative Routine-Aufgaben übernehmen, und dem Zentralisten genügend Entlastung zugunsten seiner strategischen Aufgaben zu bieten.
Welche Probleme bei der täglichen Arbeit bestehen, wird im Kapitel *Situationsbeschreibung* anhand einfacher Beispiele erläutert, hier wird auch die Motivation zur Erstellung eines solchen Systems deutlich.

2.3 Was unterscheidet EASy von einem Einsatzleitsystem ?

Ein Einsatzleitsystem "trifft" grosse Entscheidungen, ein Einsatz-Abwicklungs-System unterstützt bei den kleinen Entscheidungen: Ein Einsatzleitsystem schlägt dem Disponenten (zum Beispiel auf der Leitstelle (s.u.)) aufgrund der eingegangenen Alarmmeldung und der geographischen Lage des Einsatzes vor, welche Feuerwehr zum Einsatz gebracht werden soll[5]. Dieser Vorschlag wird vom Einsatzleitrechner erstellt und dem Disponenten als Entscheidungsgrundlage zur Verfügung gestellt. Er kann Änderungen vornehmen, die ihm aufgrund der speziellen Situation als gegeben erscheinen. Wenn EASy zum Einsatz kommt, wurde bereits alarmiert und die entsprechende Einsatzabfrage durchgeführt. Nun muß festgelegt werden, welche Rettungsmittel konkret zum Einsatz kommen. Hierzu wird aufgrund des mitgeteilten Alarmierungsstichwortes auf die sogenannte Ausrückeordnung zurückgegriffen. Die entsprechenden Einheiten werden nun zur Einsatzstelle entsandt und dem Zentralisten obliegt die Aufgabe, die eingesetzten Kräfte zu erfassen, Lagemeldungen zu protokollieren und im rückwärtigen Bereich unterstützend tätig zu werden (also Gebäudebesitzer zu informieren, den Oberbürgermeister in Kenntnis zu setzen, etc.) Und genau hier soll ihn das System unterstützen. Für die anfangs genannte Zielgruppe "kleinere" Feuerwehren macht diese Trennung Sinn, denn die benötigten Informationen für die rückwärtige Unterstützung sind hier vorhanden und nicht mehr unbedingt an der notrufannehmenden Stelle.
Noch mehr Aspekte, bei denen EASy die Arbeit erleichtern kann, sind im Kapitel *Situationsbeschreibung* aufgeführt.

[5]Hier gilt analog die Auswahl einer Abteilung bei einer größeren Feuerwehr, zum Beispiel Konstanz

3 Situationsbeschreibung

In diesem Kapitel werde ich einerseits den ”normalen“ Einsatzablauf aus Sicht des Zentralisten beschreiben und auf besondere Einsatzlagen eingehen, wobei immer das Augenmerk darauf liegt, worin in der konkreten Situation der Zentralist durch EASy unterstützt werden kann.

3.1 Der ”normale“ Einsatzablauf aus Sicht des Zentralisten

3.1.1 Vor der Alarmierung

Auf verschiedensten Meldewegen (per Telefon, per Meldung durch die Polizei oder den Rettungsdienst, etc.) kommt die entsprechende Hilfeanforderung zur Leitstelle. (Im Landkreis Konstanz ist dies die Integrierte Leitstelle in Radolfzell, die die Einsätze der meisten Feuerwehren im Landkreis und der Rettungsorganisationen koordiniert). Der Betroffene meldet an die Leitstelle die sogenannten 5 W's.

- **W**o ist die Einsatzstelle ?
- **W**as ist geschehen?
- **W**elche Art der Erkrankung oder Verletzung liegt vor?
- **W**ieviel Betroffene gibt es?
- **W**arten auf Rückfragen!

Nach dieser Meldung wird vom Disponenten der Leitstelle eine Alarmierung der jeweiligen Feuerwehr (und im Regelfall des zuständigen Rettungsdienstes) ausgelöst. Derzeit geschieht dies in der Regel per Funkalarmierung über den normalen Betriebskanal der Feuerwehren im 4m-Band (siehe Kapitel 3.3.1). In vielen Landkreisen wird hier allerdings auch schon die sogenannte digitale Alarmierung eingesetzt, bei der zusätzlich Textinformationen wie zum Beispiel der Einsatzort direkt an die Führungskräfte übertragen werden können. Im Landkreis Konstanz findet im Moment noch eine Alarmierung mit sogenannten 5-Ton-Folgen nach ZVEI-Norm statt. Daraufhin lösen bei den diensthabenden Feuerwehrleuten die

Meldeempfänger aus beziehungsweise die alarmierten Sirenen beginnen zu heulen.
In vielen Freiwilligen Feuerwehren (gerade bei hohem Einsatzaufkommen) wurde das System einer wechselnden Bereitschaft eingeführt. Damit soll eine bessere Verteilung der Einsatzbelastung auf die einzelnen Einsatzkräfte realisiert werden. Diese Feuerwehren haben ein rotierendes Alarmierungssystem aufgestellt. Dabei sind verschiedene Varianten denkbar, einige Beispiele sind nachfolgend aufgeführt:

- **Freiwillige Feuerwehr Engen:**
  Alarmierung nach Fahrzeug, das heißt einem bestimmten Fahrzeug sind bestimmte Feuerwehrleute zugeordnet, die bei Bedarf (nach Ausrückeordnung) zielgerichtet alarmiert werden.

- **Freiwillige Feuerwehr Konstanz:**
  Alarmierung nach geographischer Position des Ereignisses, das heißt die entsprechende Einheit des entsprechenden Stadtgebiets wird vollständig alarmiert.

- **Freiwillige Feuerwehr Singen, Abteilung Kernstadt:**
  Alarmierung im wöchentlichen Wechsel, das heißt die Feuerwehrleute sind in 4 sogenannte Einsatzschleifen aufgeteilt. Wenn nun zum Beispiel in Woche 1 die Einsatzschleife 1 Dienst hat, wird diese in den meisten Fällen alarmiert (Sonderfälle, s.u.). Sollte die Einsatzlage schon aufgrund der Meldung oder des Stichworts erkennbar größer sein, werden 1-2 weitere Schleifen (in diesem Falle die Schleifen 2 und 3) automatisch mitalarmiert.

- **Freiwillige Feuerwehr Singen, alle anderen Abteilungen:**
  Alarmierung ganzjährig per Funkmelder und Sirene. Tagsüber wird zusätzlich die Abteilung Kernstadt mitalarmiert, m der Situation Rechnung zu tragen, dass viele Angehörige dieser Abteilungen ausserhalb arbeiten.

- **Arbeitsschleifen:**
  In vielen Feuerwehren gibt es noch zusätzlich zu den normalen Einsatzschleifen sogenannte Arbeits- oder auch Kleineinsatzschleifen. Diese sind aus Gründen der gezielteren Alarmierung für bestimmte Einsatzlagen (Ölspurentfernung, etc.) bewußt personell geringer besetzt.

- **Sonderschleifen:**
  Zum Beispiel Sonderalarmierung der Gefahrgutgruppen bei Unfällen mit Gefahrgut. Die hier alarmierten Kräfte verfügen in der Regel über eine Sonderausbildung und müssen zur Bewältigung dieser Sonderlagen gezielt alarmierbar sein.

- **Personenrufschleifen:**
  Alarmschleifen, auf denen einzelne Personen oder Kleingruppen mit Sonderaufgaben alarmiert werden können (Kreisbrandmeister, Kommandant, Fachberater, etc.)

Diese Informationen entnimmt der Disponent beziehungsweise der Einsatzleitrechner der Leitstelle der sogenannten Alarmierungsordnung der jeweiligen Feuerwehr.

3.1.2 Nach der Alarmierung

Die Einsatzkräfte begeben sich auf die Alarmierung hin zu ihrem jeweiligen Gerätehaus. Nun fragt der erste Eintreffende bei der Leitstelle nach, welches Ereignis vorliegt und wo der entsprechende Einsatzort ist. Anschließend wird aufgrund dieser Meldung die entsprechende Ausrückeordnung ausgewählt. Die Fahrzeuge und das entsprechende Personal begeben sich zum Einsatzort und leiten die Schadensabwehr ein. Bei vielen Feuerwehren verbleibt ein Zentralist im Gerätehaus und besetzt die Funkzentrale. Er ist für die Einsatzdokumentation zuständig, ebenso für die Personal-Verwaltung, eventuelle Nachalarmierungen und generelle Unterstützung im rückwärtigen Bereich. Vor Ort ist in der Regel der Einsatzleitwagen stationiert, hier ist ein entsprechender Gegenpart zum Zentralisten zu finden, der die Geschehnisse vor Ort dokumentiert und Lagemeldungen weitergibt beziehungsweise Tätigkeiten veranlasst[1].

3.1.3 Wo kann ein Softwaresystem hier unterstützen ?

Bisher wird die Dokumentation auf Papier geführt. Dazu haben alle Feuerwehren eigene Formblätter entwickelt, die alle Informationen zum Einsatz aufnehmen. Dazu gehören: Eingesetztes Personal, Ausrückezeiten der Fahrzeuge, verwendetes Material, betroffene Personen, andere beteiligte Hilfsorganisationen, durchgeführte Maßnahmen. Diese Tätigkeit soll von EASy nun teilweise automatisiert beziehungsweise unterstützt werden.
Beim zweiten großen Aufgabenblock, nämlich der Beschaffung von Einsatzmitteln (also zum Beispiel eines Baggers, etc.) und der Information von Ämtern und Behördern (Ortspolizeibehörde, Landratsamt, Kanalbau, etc.) kann ein Softwaresystem ideal unterstützen, da hier umfangreiche Telefonlisten und Checklisten, etc. vorgehalten und bei Bedarf abgerufen werden können.

[1] In vielen Feuerwehren werden aufgrund der ansteigenden Komplexität dieser Aufgaben (dazu gehört auch die Lageerkundung, die Koordination der Zusammenarbeit mit anderen beteiligten Hilfsorganisationen) sogenannte Führungsgruppen gebildet, die nach Feuerwehr-Dienstvorschrift 100 ausgebildet wurden. Diese in der Regel personell beschränkte Gruppe ist natürlich als Zielanwendergruppe für EASy ideal.

Auch bei der Bewältigung von Einsätzen an größeren Objekten mit erhöhtem Gefahrenpotential für Mensch und Umwelt (Krankenhäuser, Radiologische Arztpraxen, Pharmazeutische Betriebe, etc.) kann durch die gezielte Hinterlegung von sogenannten Objektplänen auf besondere Punkte hingewiesen werden und der Einsatz erfolgreich durchgeführt werden.

3.2 Kommunikation mit den Einsatzkräften vor Ort

Auch die Kommunikation mit den Einsatzkräften vor Ort (Entgegennahme von Lagemeldungen, Nachforderungen, etc.) fällt in den Aufgabenbereich des Zentralisten im Gerätehaus. Auch hier kann ein Einsatz-Abwicklungs-System durch intelligente Abfragemasken unterstützen.
Desweiteren muss dokumentiert werden, wann welches Fahrzeug das Gerätehaus in Richtung der Einsatzstelle verlässt, wann es ankommt und wann es zurück ist[2].

3.2.1 Kommunikationsbedarf

Je nach Einsatz ist der Bedarf an Kommunikation unterschiedlich hoch und wichtig. Bei größeren Einsätzen, bei denen Einheiten aus vielen verschiedenen Abteilungen eingesetzt werden, um gemeinsam eine Lage zu bewältigen, ist hier ein höherer Bedarf an Kommunikation notwendig als beim Löschen eines brennenden Altpapiercontainers.

3.2.2 Kommunikationsmöglichkeiten

Gerade für die erste Situation ist es denkbar, eine Kommunikation zwischen der Einsatzstelle und dem Gerätehaus nicht nur auf einer ”verbalen“ Ebene abzuwickeln, sondern hier zusätzlich digital zu kommunizieren. Dies hilft, Verständigungsprobleme, zum Beispiel durch Störungen bei der Funkverständigung zu minimieren.

3.3 Existierende Technologien und ihre Anwendung

3.3.1 Analoger Funk

Die heute in den meisten Bundesländern eingesetzte Technologie zur Verständigung der Einsatzkräfte untereinander beziehungsweise mit rückwärtigen Stellen

[2]Dies ist zum Beispiel zur Kontrolle der Einhaltung der sogenannten Hilfeleistungsfristen notwendig und sinnvoll.

erfolgt mittels analoger Technologie. Diese Technologie wurde über die letzten Jahre perfektioniert und hat nun einen sehr hohen Standard in Beziehung auf Flächendeckung und Sprachqualität erreicht.
Es gibt im Bereich der BOS (Behörden und Organisationen mit Sicherheitsaufgaben, also Polizei, Feuerwehr, DLRG, Zoll, Technisches Hilfswerk, Rettungsorganisationen, etc.) in der Regel einen beziehungsweise mehrere Funkkanäle im 2m-Band für die Kommunikation an der Einsatzstelle und einen beziehungsweise mehrere Funkkanäle im 4m-Band für die Kommunikation mit anderen Stellen. In der Regel bleiben die Organisationen unter sich, das heißt die Polizei kommuniziert nicht direkt per Funk mit der Feuerwehr, sondern erledigt dies im persönlichen Gespräch an der Einsatzstelle beziehungsweise über die Leitstelle. Über den 4m-Funkkanal werden bei den Feuerwehren im Moment sowohl die verbale Kommunikation als auch die Alarmierung und die Kommunikation per FMS (s.u.) durchgeführt (es gibt regionale Unterschiede). Da die verwendete analoge Technologie im Vergleich zur moderneren Digitaltechnik wesentlich einfacher zu empfangen ist, kann man diesen Funkkanal mit einfachen Geräten mithören und mit entsprechenden Auswertern die übertragenen Signale auswerten.
Jede Organisation hat in der Regel noch sogenannte Ausweichkanäle, um hier bei größeren Schadenslagen oder Flächenereignissen eine Trennung vornehmen zu können.

3.3.2 ZVEI-Alarmierung

Bei der gängigen Alarmierung im 4m-Band handelt es sich um eine Folge von 5 Tönen, die vom ZVEI (Zentralverband der deutschen Elektroindustrie) normiert wurden. Die Funkmeldeempfänger der einzelnen Rettungskräfte ”lauschen“ ständig auf dem 4m-Funkkanal mit und lösen bei einer entsprechenden Dekodierung der programmierten 5-Ton-Folge aus. Nach der Alarmierung folgt der sogenannte Kanalbelegton, der andere davon abhalten soll, während der Alarmierung zu funken, danach folgt in der Regel eine Durchsage, die je nach Vereinbarung mit der alarmierten Feuerwehr bereits das Einsatzstichwort und die Einsatzstelle enthält.
Die 5 Ziffern die in diesem System übertragen werden, folgen einem bestimmten System, wodurch Fehlalarmierungen bei Überreichweiten vermieden werden sollen. Die Belegung der ersten zwei Ziffern ist bundesweit einheitlich nach Regionen abgestimmt, z.B. steht die 1 an erster Stelle für Baden-Württemberg. Im Landkreis Konstanz beginnen die Schleifen (man könnte auch ”Rufnummern“ dazu sagen) der Feuerwehren generell mit 16...

3.3.3 Digitale Alarmierung

Die digitale Alarmierung findet im Gegensatz zur ZVEI-Alarmierung auf einem getrennten Funkkanal statt. Um die digitale Alarmierung einzuführen, muss im jeweiligen Bereich daher eine zusätzliche Infrastruktur geschaffen werden, um eine Flächendeckung zu erreichen. Bei der digitalen Alarmierung ist eine noch gezieltere Alarmierbarkeit gegeben, da es nicht mehr die Restriktion auf 1000 Rufadressen gibt. Desweiteren ist es möglich, digital das Einsatzstichwort und den Einsatzort zu übertragen, Informationen die bei einem entsprechend ausgestatteten Meldeempfänger dann zum Beispiel den Führungskräften gleich zur Verfügung stehen.

Nach dem selben Verfahren ist es nach einer Einführung der digitalen Alarmierung[3] dann später möglich, zum Beispiel einen Einsatz der von der Leitstelle aufgenommen wird direkt in das Einsatz-Abwicklungs-System zu übernehmen.

3.3.4 FMS - Funk-Melde-System

Die Spezifikation des Funk-Melde-Systems beschreibt ein digitales Telegramm, das über den Funkkanal der jeweiligen Organisation übertragen wird. Ursprünglich von der Polizei eingeführt wurde dieses System auch schnell vom Rettungsdienst und den Feuerwehren übernommen und findet seither eine große Verbreitung. Das System dient hauptsächlich dazu, den Funkkanal zu entlasten, da viele Funksprüche und Routinemeldungen sich hier auf einen Tastendruck und eine kurze Datenübertragung reduzieren lassen.
Als Beispiel soll folgender exemplarischer Funkspruch dienen: ”Florian Singen von Florian Singen 1/11 kommen“ – ”Hier Florian Singen, kommen.“ – ”Abfahrt zur Einsatzstelle“ – ”Verstanden, kommen“ – ”Ende“
Dieses Gespräch lässt sich mit FMS auf ca. 300ms verkürzen. Das FMS-Telegramm folgt einem einheitlichen Schema und kann neben der jeweiligen Kennzahl des Fahrzeugs auch noch eine sogenannte taktische Kurzinformation (TKI) und ein Folgetelegramm enthalten, in dem zum Beispiel der Einsatzort auf das Display im Fahrzeug übertragen wird. Die Auswertung der FMS-Telegramme kann ein handelsüblicher Computer mit einer Soundkarte und entsprechender Software übernehmen, alternativ kann auch dedizierte Hardware dazu in Einsatz gebracht werden.

[3] In vielen Landkreisen, die ein entsprechendes System noch nicht eingeführt haben, wird die digitale Alarmierung aufgrund der angespannten Haushaltssituation öffentlicher Kassen noch lange auf sich warten lassen

3.3.5 In der Zukunft: Digitaler Funk

Ursprünglich sollten bis zum Jahr 2006 die reservierten Frequenzen im analogen Funknetz an die Regulierungsbehörde für Telekommunikation und Post (RegTP) zurückgegeben werden. Dies macht die Einführung eines alternativen Systems notwendig. Im Hinblick auf die Fußball-Weltmeisterschaft 2006 in Deutschland sollte das System bis dorthin einsatzbereit sein. Aufgrund technischer und philosophischer Differenzen und trotz diverser Beschlüsse der Innenministerkonferenz konnte allerdings bis zum heutigen Tag keine Entscheidung für die Einführung eines bestimmten Systems getroffen werden. Problematisch beim digitalen Funk ist die Flächenabdeckung, da hier mit relativ kleinen Zellen gearbeitet werden muss und daher das Funknetz entsprechend teuer in Anschaffung und Unterhalt ist. Im digitalen Funksystem sollen dann nach gängiger Meinung sowohl FMS als auch die digitale Alarmierung aufgehen und so ein universelles Funknetz geschaffen werden, das eine behördenübergreifende Kommunikation ermöglicht. Wann und ob dieses Funksystem überhaupt zum Einsatz kommen wird ist noch unklar, jedoch ist aufgrund der offenen Softwarearchitektur von EASy eine Integration später problemlos möglich.

3.3.6 Kommunikation der Zukunft: Mesh Networks ?

Neu in der Diskussion um flächendeckende Netzwerk-Versorgung sind die sogenannten Mesh Networks. Dabei handelt es sich um die im Prinzip simple Idee, jedes Gerät, das an einem solchen Netzwerk teilhaben soll, nicht nur als Empfänger auszulegen, sondern als Verstärker zu benutzen, um die Reichweite zu erhöhen. Dazu ist von der Industrie angedacht, Fahrzeuge, Hausinstallationen und mobile Endgeräte als Repeater zu benutzen.
Die Nutzung dieser Technologie zur Kommunkation zwischen der/den Einsatzstelle(n) und den jeweiligen Führungsstellen ist bei entsprechender Verfügbarkeit sicherlich interessant, da hier eine weitgehende Unabhängigkeit von einer stationären, organsierten Infrastruktur vorliegt.(SR03)

3.4 Besondere Einsatzlagen

Eigentlich sind alle Feuerwehreinsätze besondere Einsatzlagen, denn ein Patentrezept gibt es nie. Was es jedoch gibt, sind Standardsituationen, die keine besondere Planung oder Organisation benötigen, da die Lage einerseits überschaubar und nicht besonders personal- und materialintensiv ist.

Neben diesen Standard-Situationen, in denen EASy zum Einsatz kommen soll um das Protokoll zu führen, gibt es Situationen, in denen EASy besonders

unterstützen kann. Nachfolgend seien einige dieser Situationen stellvertretend genannt.

3.4.1 Großschadenslage

Bei einer sogenannten Großschadenslage[4] unterstützt EASy den Zentralisten dadurch, dass hier Routinetätigkeiten, wie die Erfassung der FMS-Status automatisch vorgenommen und dem aktuellen Einsatz zugeordnet werden. Ebenso wird durch die Führung eines elektronischen Einsatzprotokolls im Netzwerkbetrieb der Informationsfluß verbessert.

3.4.2 Flächenereignis

Bei einem Flächenereignis[5] ist es besonders wichtig, den Überblick über die offenen Einsatzstellen zu behalten und eine Protokollierung der Einsätze (zu Abrechnungszwecken) zu gewährleisten. Da hier die Einsätze in der Regel nicht nur parallel sondern auch sequentiell abgearbeitet werden, wurde speziell für diese Situation die Möglichkeit geschaffen, Einsätze ohne Ausrückevorschlag anzulegen und der Reihe nach zu bearbeiten. Ebenso unterstützt die farbliche Markierung der Einsätze in der Übersicht den Zentralisten bei der Durchführung seiner Tätigkeit. Auch wurde eine Funktion implementiert, die eine automatische Übergabe eines Einsatzmittels in einen anderen Einsatz ermöglicht, sobald von diesem Einsatzmittel eine entsprechende Meldung (zum Beispiel per FMS) ankommt.
Sowohl für die Großschadenslage als auch das Flächenereignis gilt, dass die Netzwerkfähigkeit und der ständige Gleichstand der Informationen auf allen Arbeitsplätzen eine gezielte Auslagerung von Funktionen und Kompetenzen ermöglicht (zum Beispiel Bildung eines Lagezentrums, das die Einsätze priorisiert und zur Abarbeitung weiterleitet.) Bisher konnten diese Tätigkeiten auf Papier nur sehr mühsam und mit großer Redundanz erledigt werden.

3.4.3 Automatische Brandmeldeanlage

Zahlreiche Objekte und Gebäude sind aus baurechtlichen oder versicherungsrechtlichen Gründen mit einer Brandmeldeanlage ausgestattet, die automatisch bei Vorliegen einer erkannten Gefahrensituation Kontakt zur Feuerwehr herstellt. Gerade bei einem nächtlichen Auslösen einer Brandmeldeanlage ist der Besitzer beziehungsweise verantwortliche Mitarbeiter in der Regel nicht über

[4] Massenanfall von Verletzten, Großbrand auf einem Bauernhof mit unzureichender Wasserversorgung, etc.

[5] zum Beispiel Hochwasser/Starkregen mit vielen (bis zu hunderten) Einsatzstellen gleichzeitig

das Ereignis informiert. Diese Personen erhalten hier eine Mitteilung von der Feuerwehr. Hierbei unterstützt das System durch hinterlegte Objektpläne, Telefonlisten und Checklisten für das jeweilige Objekt. Diese Informationen, die sonst mühsam aus mehreren Quellen zusammengetragen werden müssen, stehen nach dem Anlegen eines Einsatzes mit der entsprechenden Adresse sofort zur Verfügung.

3.4.4 Katastropheneinsatz

Auch im Katastropheneinsatz bei dem auf jeden Fall mehrere Hilfsorganisationen Hand in Hand arbeiten und der sich über einen längeren Zeitraum erstreckt, unterstützt das System bei der Führung eines Protokolls und hilft so, alle mit dem Einsatz der jeweiligen Feuerwehr betrauten Zentralisten auf dem gleichen Informationsstand zu halten. Auch ist bei einem Einsatz von EASy denkbar, diese Systeme zu vernetzen und so von einer zentralen Stelle aus Zugriff auf Informationen aller entsprechend ausgerüsteten Feuerwehren zu erhalten.

3.5 Neue Statistik-Möglichkeiten mit EASy

Durch die digitale Erfassung der Einsatzprotokolle ist es beim Einsatz von EASy leichter als zuvor, Statistiken und Auswertungen über verschiedenste wichtige Aspekte zu erstellen. Hier seien stellvertretend Auswertungen genannt, wie viele Einsatzkräfte zu welcher Uhrzeit zur Verfügung stehen. Dadurch wird den Führungskräften eine höhere Planungssicherheit gegeben. Auch können nun für die einzelne Einsatzkraft Statistiken erstellt werden. Exemplarisch ist hier zum Beispiel die Jahresbelastung der einzelnen Einsatzkraft durch Einsätze zu nennen.

4 Software

4.1 Entwicklungsumgebung und -entscheidungen

4.1.1 Eingesetzte Software

4.1.1.1 Visual Studio .NET

Entwickelt wurde EASy unter der .NET-Technologie von Microsoft[1]. Unter vielen Alternativen habe ich diese Technologie ausgewählt, um mich einerseits in diese kommende Technologie und die damit verbundenen Entwicklungsparadigmen einzuarbeiten, andererseits auch, weil das .NET-Framework viele Funktionen bereitstellt, die ansonsten von Hand nachprogrammiert oder zugekauft werden müssen. So blieb mehr Zeit für die eigentliche Entwicklung. Die für mich ideale Entwicklungsumgebung war Visual Studio .NET von Microsoft, da es durch einen grafischen GUI-Editor bei der Entwicklung unterstützt und durch Tools wie den Objektbrowser bei der Verwaltung eines großen Softwareprojektes wie es EASy darstellt, hilft, den Überblick zu behalten.

4.1.1.2 Windows 2000/2003 Server

Als Grundlage für das Testsystem wurde ein Windows 2000-Server verwendet, da der Windows 2003-Server zu Beginn der Entwicklung noch nicht verfügbar war. Die Wahl fiel auf dieses System, da einerseits ein Windows-System die Servergrundlage bilden sollte, um die eingesetzte .NET-Technologie vollständig zu unterstützen, andererseits weil die verwendete Messaging-Software (s.u.) von Windows 2000 bereits mitgeliefert wird.
Prinzipiell ist das System auch vollständig auf einem mit Windows 2000 Professional oder Windows XP Professional ausgestatteten Computer zu verwenden, wobei es hier Einschränkungen bei der Rechtevergabe und der maximalen Computeranzahl im Verbund gibt. Unter Windows 2000/2003-Server sind diese Beschränkungen zumindest in der Theorie aufgehoben, die eingesetze Hardware limitiert gleichwohl die maximale Benutzeranzahl.

[1] http://www.microsoft.com/net

4.1.1.3 MySQL

Bei MySQL handelt es sich um einen Datenbankserver der schwedischen Firma MySQL A.B.[2] Der Datenbankserver unterstützt die meisten Kommandos der ANSI-Spezifikation 99 und wird derzeit von ca. 75 Entwicklern(@My03a) weltweit weiterentwickelt. Das System steht unter bestimmten Bedingungen unter der GPL[3] und hat in den letzten Jahren eine weltweite Nutzer-Gemeinde bekommen. Im Jahr 2003 hat MySQL A.B. einen Vertrag mit SAP geschlossen, wonach das Produkt SAP DB in Zukunft von MySQL A.B. unter dem Namen MaxDB weiterentwickelt wird.(@My03a)
Aufgrund der großen Verbreitung und guten Verfügbarkeit von MySQL habe ich mich dazu entschlossen, diese Software als Datenbankgrundlage für EASy einzusetzen. Natürlich können die Datenbanken nahezu unverändert in ein anderes Datenbanksystem (zum Beispiel Microsoft SQL-Server, Oracle, etc.) übernommen werden,[4]. Um die Datenbank bis auf die genannten Einschränkungen portierbar und übersichtlich zu halten, wurde auf die Einführung von Constraints[5] verzichtet. Die Pflicht zur Einhaltung der referentiellen Integrität liegt daher bei den entsprechenden EASy-Komponenten (Server, Editor).
MySQL hat sich als sehr performant herausgestellt(@My03b), ist gut skalierbar und kann eine plattformübergreifende Datenbankreplikation durchführen, ist also für den Einsatz in EASy eine hervorragende Wahl.
Um die Verwaltung der Datenbank und die Veränderungen an den Tabellen zu vereinfachen wurde die Software MySQL-Studio von NAVICAT(@Pr03) angeschafft. Damit war es mir möglich, während der Entwicklung auftretende Fehler (zu kleine Felder) leicht zu korrigieren.

4.1.2 Eingesetzte Sprachen/Technologien

4.1.2.1 C#

Im Juni 2000 kündigte Microsoft neben der .NET-Plattform auch eine neue Programmiersprache an: C#.
C# (gesprochen: C Sharp) wurde mit großem Augenmerk auf die ideale Unterstützung der .NET-Funktionalitäten entwickelt, obgleich .NET mit vielen verschiedenen Sprachen hervorragend harmoniert.
Bei der Entwicklung von C# wirkte Anders Heljsberg (der „Erfinder“ von Borland's Delphi) maßgeblich mit. Er und seine Kollegen nahmen die in Ihren Au-

[2] Im Internet zu finden unter http://www.mysql.com
[3] GNU General Public License, die Lizenz unter der viele Open-Source-Projekte veröffentlicht werden
[4] Die Datenbank muss einen ODBC-Treiber anbieten und die LAST_INSERT-Funktion unterstützen
[5] Regeln, die zum Beispiel das Löschen von referenzierten Daten verhindern

gen besten Technologien und Methoden aus verschiedenen Programmiersprachen und faßten die Vorteile dieser Technologien in C# zusammen. C# weist starke Ähnlichkeiten zu Java und C++ auf, weicht aber in einigen Aspekten auch davon ab.
Einige Gemeinsamkeiten mit Java sind:

- Es wird in eine maschinenunabhängige Zwischensprache kompiliert, die in einer kontrollierten Umgebung ausgeführt wird (das heißt die Software ist plattformunabhängig, sofern eine Version dieser Laufzeitumgebung für das Zielsystem vorhanden ist)
- Eine Garbage Collection wird automatisch durchgeführt
- Pointer sind weitgehend eliminiert
- Keine Header-Dateien, alle zusammengehörigen Dateien sind in Packages organisiert
- Alle Klassen stammen vom „Ur"-Objekt **Object** ab
- Volle Thread-Unterstützung
- Definition von Schnittstellenklassen (Interface) möglich, Klassen können mehrere Interfaces erben, keine Mehrfachvererbung von Klassen vorgesehen
- Eine automatische Initialisierung von Werten wird vom Compiler durchgeführt

Ein Ziel bei der Entwicklung von C# war unter anderem, die Sprache so einfach wie möglich zu gestalten. Der größte und auffälligste Unterschied zwischen C# auf der einen und C beziehungsweise C++ auf der anderen Seite ist, dass mit C# entwickelte Programme ohne erneute Kompilierung und Anpassungen weitgehend plattformunabhängig sind. Das heißt, ein Programm läuft nicht nur unter dem Betriebssystem, unter dem es entwickelt wurde, sondern auf allen Systemen, für die eine Version des .NET-Framework existiert.
Dabei müssen diese Systeme nicht unbedingt PCs sein, es kann sich dabei beispielsweise auch um tragbare Geräte wie PDAs oder Mobiltelefone handeln. So existiert zum Beispiel eine speziell an die Eigenschaften mobiler Endgeräte angepasste, besonders kompakte Version des .NET-Frameworks. Ein ähnliches Konzept verfolgt die Sprache Java mit ihren verschiedenen Editionen (J2SE (Java 2 Platform, Standard Edition), J2ME (Java 2 Platform, Micro Edition), J2EE (Java 2 Platform, Enterprise Edition)), womit ebenfalls plattformunabhängige Programme erzeugt werden können.

Ergänzt wird C# durch die ebenfalls durchgängig objektorientierte Klassenbibliothek des .NET-Framework, welche im Vergleich zu den anderen häufig eingesetzten Klassenbibliotheken wie MFC oder JFC deutlich übersichtlicher gestaltet ist.(@Go03)
Ich habe die Sprache C# zur Entwicklung von EASy aus zweierlei Gründen verwendet: Einerseits hat mich das (zugegebenermaßen nicht sonderlich neue) Konzept des .NET-Frameworks überzeugt, andererseits wollte ich für zukünftige Projekte Erfahrung mit einer objektorientierten Sprache sammeln, da ich bisher hauptsächlich mit ereignis-orientierten Sprachen entwickelt habe.
Ich habe mir daraufhin die meisten Sprachen des .NET-Frameworks (VisualBasic.NET, etc.) angesehen und mich dann für die in meinen Augen am besten strukturierte, nämlich C#, entschieden.

4.1.2.2 Microsoft Message Queueing

Bei der Entwicklung verteilter Anwendungen entsteht oftmals die Situation, dass auf die Bearbeitung einer Anforderung nicht gewartet werden muss, sondern der Programmfluss sofort weitergeführt werden kann, oder dass eine Bearbeitung zu lange dauert, um direkt darauf zu warten. Zu diesem Zweck wurden Messaging-Systeme entwickelt, die Nachrichten zwischen den Applikationen beziehungsweise Schichten der Applikation transportieren und für die sichere Übertragung sorgen.
Microsoft Message Queueing (kurz: MSMQ) ist solch ein Messaging-System, das Applikationen erlaubt, Daten asynchron auszutauschen. Dazu bedienen sich Messaging-Systeme in der Regel sogenannter Queues, in denen die Nachrichten bis zur Verarbeitung gesammelt werden. Der verarbeitende Dienst holt dann die für ihn bestimmten Nachrichten aus der Queue und verarbeitet sie entsprechend.(@Mi03d)

Im Rahmen der Software-Entwicklung habe ich mehrere Messaging-System evaluiert (MSMQ, MsgConnect(@El03), Jabber(@Ja03)) und mich dann schlussendlich für MSMQ entschieden, da die entsprechenden Zugriffs-Funktionen bereits im .NET-Framework integriert sind. Allerdings wurde die komplette Messaging-Funktionalität in ein eigenes Modul von EASy ausgelagert, was es ermöglicht, MSMQ durch ein anderes Messaging-System zu ersetzen.
MSMQ bietet zwei verschiedene Betriebsarten, je nach Plattform, auf der das System zum Einsatz kommt. Wenn MSMQ auf einem Windows 2000/2003-Server mit installiertem Active Directory zum Einsatz kommt, ist es möglich, Nachrichten über mehrere Computer zu routen, bei der Installation und dem Betrieb unter einem der Microsoft Client-Betriebssysteme (Windows NT 4, Windows 2000 und Windows XP Prof.) ist dies nicht möglich, der Betrieb also auf einen einzigen Messaging-Server beschränkt.

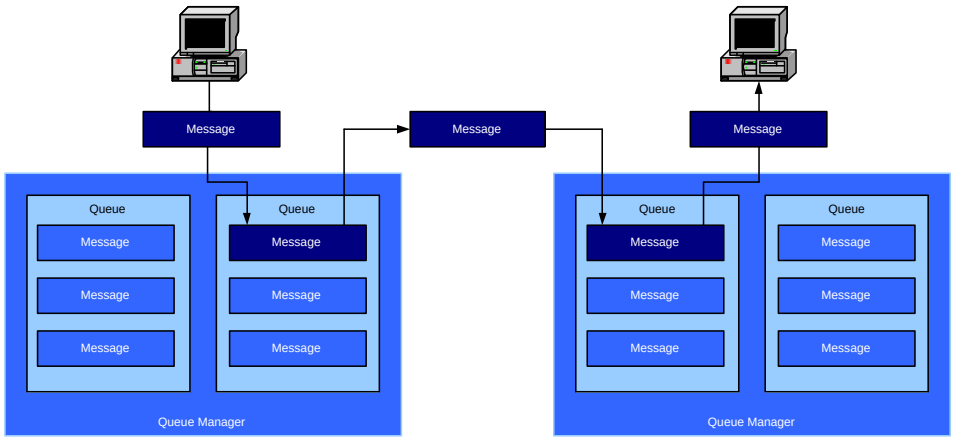

Abb. 4.1: Beispiel: Transfer einer Nachricht über mehrere Queue-Manager

Getestet wurde diese Funktionalität nicht, da im momentanen Entwicklungsstadium von EASy dafür kein Bedarf besteht. Was getestet wurde ist die Offline-Funktionalität, also die Möglichkeit, bei unterbrochener Verbindung trotzdem Nachrichten zu senden. Diese werden dann bei Wiederherstellung der Verbindung verarbeitet.

4.1.2.3 SQL

SQL, die Structured Query Language, ist eine Abfragesprache für Datenbankinhalte, die 1986 als Industriestandard festgelegt wurde. Seither hat SQL viele Erweiterungen erfahren (Objektrelationale Datenbanken, XML-Unterstützung), die Urform hat allerdings bis heute Gültigkeit und nahezu kein modernes Programm, das Zugriff auf eine Datenbank benötigt, kommt ohne SQL aus.
Da bei der Entwicklung von EASy wie bereits oben erwähnt, auf die Nutzung von datenbankspezifischen Eigenheiten weitgehend verzichtet wurde, ist auch die verwendete SQL-Syntax bewusst einfach gehalten und verwendet ausschließlich die Kommandos SELECT, INSERT und UPDATE in einfachen Formen.

4.1.2.4 TeX

TEX (sprich: „Tech“) ist ein Satzsystem, das Donald E. Knuth (Stanford University) im Laufe von 10 Jahren entwickelte. Es ist insbesondere für die Erstellung wissenschaftlicher Veröffentlichungen, die mathematische Formeln enthalten, geeignet. TEX ist im engeren Sinne eher eine plattformunabhängige Programmiersprache für Dokumente, denn ein Textverarbeitungssystem. Eine Erweiterung erfuhr TEX durch die LATEXErweiterungen, die die auf Drucksatz ausgerichteten Funktionen von TEX um logische Strukturen erweitern, die dem Autor die Möglichkeit geben, Inhalt und Darstellung voneinander zu trennen (Gün02).

Diese Fähigkeiten machen TEX zum prädestinierten Ausgabe-System für EASY. Ein TEX-Dokument aus einer oder mehreren Dateien, die alle im ASCII-Format vorliegen und daher leicht automatisiert erstellt werden können. TEX wird in EASY zum Beispiel verwendet, um Berichte und Formulare formatiert auszugeben. Dabei kümmert sich das Satzsystem TEX bei entsprechender Einrichtung selbständig um Seitenumbrüche und sonstige satztechnische Feinheiten, die ansonsten mühsam berechnet werden müssten.
Auch diese Diplomarbeit ist in TEX verfasst, benutzt wurde hierzu als Editor WinEdt 5.3[6], als TEX-Grundlage wurde MikTeX[7] unter Windows eingesetzt.
Ein TEX-Dokument verwendet sogenannte Tags um Formatierungen und Steuerfunktionen einzuleiten. Ein Tag beginnt immer mit einem Backslash (\) und kann bis zu 9 Parameter erhalten. Viele Pakete existieren, die Tags und Makros für Standardaufgaben mitbringen (Grafikeinbindung, Tabellensatz, Buchsatz, automatische Inhaltsverzeichnisse, etc.).
Das folgende TEX-Dokument (@La03)

Quellcode 4.1: Beispiel für ein TEX-Dokument

```
\documentclass[openright,twoside]{book}
\usepackage[german]{varioref}

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam
nonummy nibh euismod tincidunt ut \textsc{laoreet dolore} magna
aliquam erat \textbf{volutpat. Ut wisi enim} ad minim veniam, quis
nostrud exerci tation ullamcorper \textit{suscipit} lobortis nisl
ut aliquip ex ea commodo consequat.\\

Duis autem vel eum \texttt{iriure dolor} in hendrerit in vulputate
velit esse molestie consequat, vel illum dolore eu feugiat nulla
facilisis at vero et accumsan et iusto odio dignissim qui blandit
praesent luptatum zzril delenit augue duis dolore te feugait nulla
facilisi.
```

[6] http://www.winedt.com
[7] http://www.miktex.org

ergibt in der Ausgabe folgendes:

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut LAOREET DOLORE magna aliquam erat **volutpat. Ut wisi enim** ad minim veniam, quis nostrud exerci tation ullamcorper *suscipit* lobortis nisl ut aliquip ex ea commodo consequat.

Duis autem vel eum `iriure dolor` in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi.

4.1.2.5 XML

XML, die Extensible/Extended Markup Language, beschreibt eine Sprache, die es ermöglicht, Daten in einer strukturierten Form abzulegen und weiterzuverarbeiten. Im Gegensatz zu HTML wird in einem XML-Dokument der Inhalt der Information und deren Struktur in den Vordergrund gestellt, bei der Verwendung von HTML (zumindest in der heutzutage üblichen Anwendungsform) ist die Darstellung auf dem Bildschirm das maßgebliche Ziel. Bei HTML ist zwar im weiteren Sinne auch eine Strukturierung von Daten möglich, jedoch nur in Form verschiedener Gliederungsebenen, während es mit XML möglich ist, die Daten so abzulegen, wie es Ihrem Inhalt entspricht. XML-Dokumente sind daher in der Regel sowohl für den Menschen als auch für Maschinen lesbar.
XML ist ein offener Industrie-Standard, dessen Entwicklung von einem unabhängigen Gremium, dem World Wide Web Consortium[8] überwacht und vorangetrieben wird.
Rund um XML ist eine ganze Familie von Sprachen entstanden, die zum Zweck haben, den Einsatz von XML optimal zu ermöglichen. Die wichtigsten Vertreter sind:

- **XML Schema**: wird zur Beschreibung der Form und des Inhalts einer XML-Struktur verwendet. Mit XML Schema können XML-Daten validiert werden.
- **DTD (Document Type Definition)**: Der Vorgänger von XML Schema, ebenfalls eine Sprache zur Beschreibung der Struktur eines XML-Dokuments, hat inzwischen allerdings größtenteils an Bedeutung verloren beziehungsweise wurde durch XML Schema ersetzt.
- **XSL (Extensible Stylesheet Language)**: Umfaßt die Sprachen:

[8] http://www.w3c.org/XML/

- **XSLT (XSL Transformation)**: Wird zur strukturierten Umwandlung von XML-Daten in andere Formate verwendet. Denkbar sind Umwandlungen von XML in XML, von XML in HTML und viele andere Varianten. In EASy wird XSLT eingesetzt, um XML in ein TEX-Dokument umzuwandeln.
- **XPath (XML Path Language)**: Wird verwendet um bestimmte Teile eines XML-Dokuments gezielt zu adressieren. Bei der oben erwähnten Transformation von XML in ein TEX-Dokument wird extensiv Einsatz von XPath betrieben.
- **XSL-FO (XSL Formatting Objects)**: Eine Sprache, die die Druck-Ausgabe von XML-Dokumenten in Zusammenhang mit den beiden anderen Sprachen aus der XSL-Familie vereinfachen soll.

Aufgrund der Vorteile die XML dank dieser Sprachenvielfalt bietet, war die Entscheidung klar, für die Kommunikation im EASy-System auf XML zu setzen, vor allem auch, weil eine XML-basierte Kommunikation aufgrund vorhandener Tools und Funktionsbibliotheken einfach zu implementieren ist und das .NET-Framework hier entsprechende Methoden, zum Beispiel zur Serialisierung[9] anbietet.
Es wurde eine einheitliche Struktur für die verwendeten Nachrichten geschaffen, die die Einführung einer Validierung der Nachrichten ermöglicht.
Ein Beispiel für eine solche Nachricht sieht wie folgt aus:

Quellcode 4.2: Beispiel für eine XML-Nachricht

```
<EASy>
  <EASyCommand name="FMSSignal">
    <dataFMS xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <FMSKennung>62665110</FMSKennung>
      <DatumUhrzeit>10.08.2003 15:35:09</DatumUhrzeit>
      <Status>4</Status>
      <Tki>1</Tki>
      <Richtung>0</Richtung>
      <Baugruppe>1</Baugruppe>
      <Folgetelegramm />
      <quittiert>true</quittiert>
      <FahrzeugID>0</FahrzeugID>
      <Quelle/>
```

[9] Umwandlung eines Objekts aus seiner im Speicher vorhandenen Form in eine speicherbare/übertragbare Form, in diesem Fall XML

```
      <EinsatzID>0</EinsatzID>
16    </dataFMS>
  </EASyCommand>
18 </easy>
```

Bei diesem Beispiel handelt es sich um eine einfache Nachricht, in der alle notwendigen Felder eines FMS-Signals übertragen werden. Der Server führt nach Empfang dieser Nachricht zum Beispiel folgende Tätigkeiten aus: Schreiben des FMS-Signals in die Datenbank, Überprüfung, ob das Fahrzeug mit der FMS-Kennung `62665110` in einem Einsatz verwendet wird (dazu später mehr), informiert die angeschlossenen Clients über die Statusänderung und führt weitere Verwaltungsfunktionen durch.
Durch den konsequenten Einsatz von XML für die Kommunikation ist die Offenheit des Systems gegeben, um Daten für andere Systeme zur Verfügung zu stellen, beziehungsweise von Fremd-Systemen Daten zu übernehmen.

4.1.3 Warum gerade diese Kombination ?

Letztendlich hat sich diese Kombination für mich als bestmögliche Kombination herausgestellt, was nicht bedeuten soll, dass man es nicht anders oder besser machen kann. Ich konnte jedoch bei der Erarbeitung dieser Diplomarbeit viel im Umgang mit für mich teilweise neuen Technologien (.NET-Framework, C#) lernen. Da ich im Rahmen dieser Diplomarbeit keinerlei Vorgaben hatte, war es auch interessant, verschiedene Technologien zu evaluieren, Ansätze zu verwerfen und andere Ansätze fortzusetzen.
Die eingesetzten Technologien ergänzen sich gut, was nicht zuletzt daran liegt, dass sie größtenteils vom gleichen Hersteller stammen. Die Nicht-Microsoft-Bestandteile sind alle so ausgewählt, dass eine bestmögliche Kompatibilität gegeben ist.

4.1.4 Multi-Plattform-Einsatz

Im Abschnitt über das .NET-Framework und C# ist von Plattformunabhängkeit die Rede. Auch ich habe bei der Entwicklung unter anderem deswegen das .NET-Framework ausgewählt. Microsoft versteht unter Plattformunabhängigkeit allerdings im Moment die Tatsache, dass .NET-Programme auf verschiedenen Prozessoren und in verschiedenen Umgebungen (Windows Mobile - TabletPC, PocketPC, etc.) funktionieren.

4.1.4.1 Das MONO-Projekt

Da die Spezifikation des Frameworks offen und der Funktionsumfang bekannt ist, hat sich die Ximian Inc.[10] dazu entschlossen, im Rahmen des Open-Source-Projektes MONO das .NET-Framework auf Linux zu portieren. Das Projekt ist weit fortgeschritten, die Ergebnisse können auf der MONO-Website[11] begutachtet werden. Da Programme die auf dem .NET-Framework basieren in einer sogenannten Intermediate Language (IL) vorliegen, kann mit MONO die gleiche .EXE-Datei auf allen unterstützten Plattformen verwendet werden.

4.1.4.2 Probleme bei der Portierung

Meine Versuche, EASy unter Linux und MONO zum Funktionieren zu bewegen haben gezeigt, dass dies im Moment (August 2003) noch nicht möglich ist. Teilmengen von EASy funktionieren, viele der bei der Entwicklung verwendeten Funktionen sind jedoch noch nicht in MONO realisiert. Ich bin allerdings sicher, dass bis Ende des Jahres 2003 große Teile von EASy zum Beispiel auch auf einer Linux-Plattform funktionieren werden.
Da das Microsoft Message Queueing nicht ohne weiteres portiert werden kann, und von Microsoft zumindest in der nächsten Zeit keine Portierung zu erwarten ist, muss dann hier ein entsprechender Ersatz zum Einsatz kommen. Aussichtsreicher Kandidat ist hier die Software IBM WebSphere MQ[12], die jedoch bei der Entwicklung mangels einer Lizenz dieser Software nicht getestet werden konnte. Dieses IBM-Produkt unterstützt viele Plattformen (u.a. Linux, Windows, OS/2, AIX, etc.)(@IB03) und ist weitgehend mit der Funktionalität von Microsoft Message Queueing deckungsgleich.
Eine Portierung von EASy auf andere Plattformen ist also eine reine Frage der Zeit.

[10] http://www.ximian.com
[11] http://www.go-mono.com
[12] früher: IBM MQSeries

4.2 Software-Architektur

Nach dieser Einführung in die verwendeten Technologien möchte ich nun auf das entstandene System EASy näher eingehen.

4.2.1 System-Struktur

Zentrale Komponente von EASy ist der EASy-Server, der über verschiedene Message Queues mit den Clients und anderen Geräten kommuniziert. (siehe Abbildung 4.2)
Für den Hauptserver gibt es 2 Queues: **ServerQueueIn** und **ServerQueueOut**. In die Queue **ServerQueueIn** werden von den Clients Nachrichten gestellt, die der Server anschließend verarbeiten soll. Ein Beispiel für eine solche Nachricht ist eine Straßenabfrage[13].

Quellcode 4.3: Beispiel für eine XML-Nachricht - Straßenabfrage

```
<EASy commPartnerName="EASyWindowsClient1">
  <EASyCommand name="GetStrassenByName">
    <Strassenname>Post</Strassenname>
  </EASyCommand>
</EASy>
```

Diese Nachricht wurde vom Client **EASyWindowsClient1** in die Queue **ServerQueueIn** gestellt. Der Server verarbeitet daraufhin diese Nachricht, er sucht also alle passenden Straße heraus. Die Antwort auf die Anfrage lautet daher:

Quellcode 4.4: Beispiel für eine XML-Nachricht - Straßenabfrage-Antwort

```
<EASy>
  <EASyReply command="getStrassenByName">
     <Strasse StrassenID="649" Ort="Singen"
       ZustaendigerStandortID="SIKE">
       Am Posthalterswaeldle
     </Strasse>
     <Strasse StrassenID="926" Ort="Singen-Friedingen"
       ZustaendigerStandortID="SIFR">
       Postweg
     </Strasse>
  </EASyReply>
</EASy>
```

[13] In EASy ist eine Datenbank vorhanden, die Straßen zu den verschiedenen Standorten einer Feuerwehr zuordnet. Bei der Eingabe eines Straßennamens, zum Beispiel beim Anlegen eines neuen Einsatzes, wird eine Suche in dieser Datenbank durchgeführt, die alle Straßen auf die das Suchmuster zutrifft zurückgeliefert.

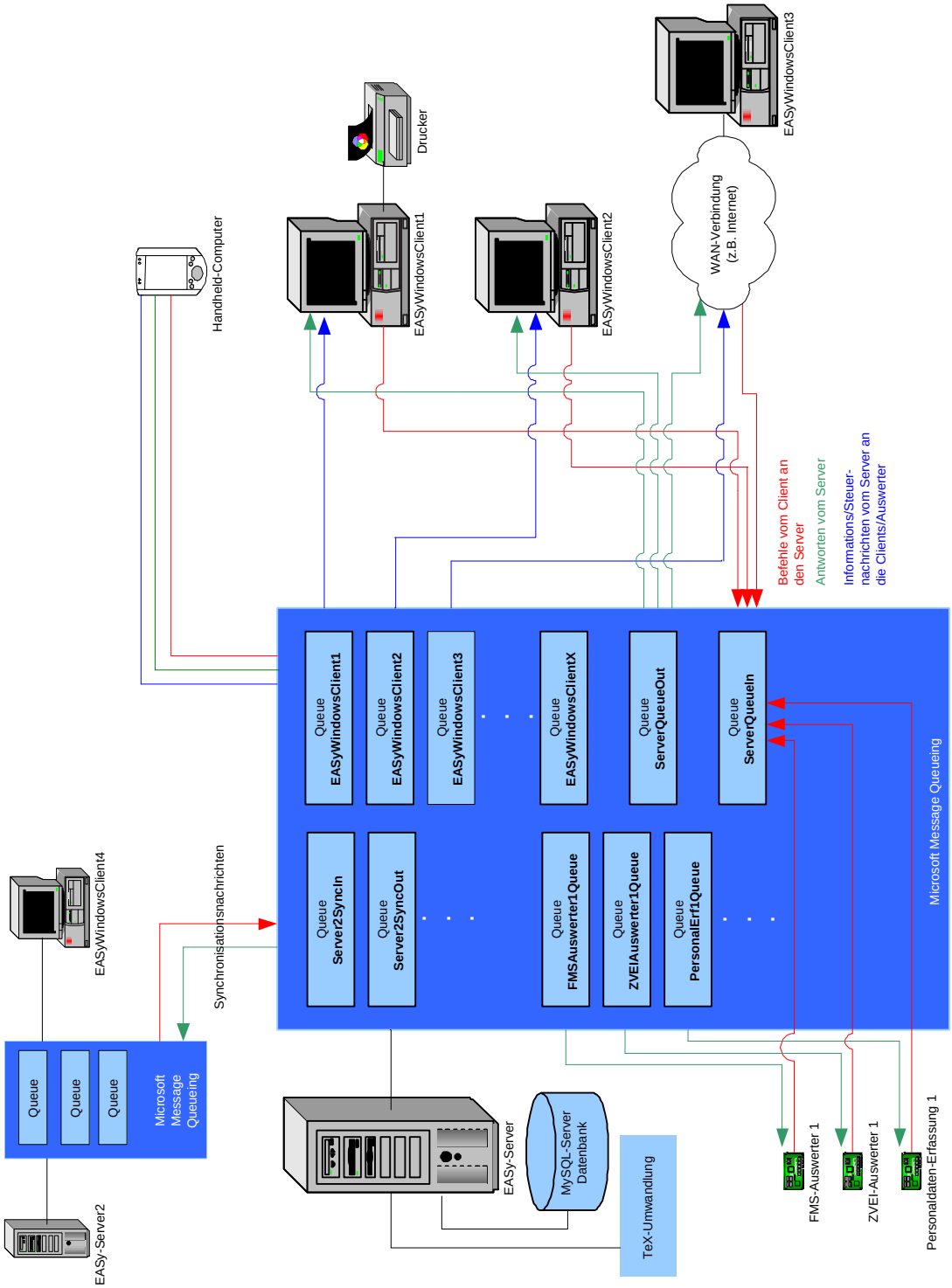

Abb. 4.2: Die Struktur des EASy-Systems

In dieser Antwort sind alle für den Client notwendigen Daten enthalten, um eine Liste der in Frage kommenden Straßen anzuzeigen. Dabei ist das System in der Lage, bei nur einer gefundenen Straße diese entsprechend gleich auszuwählen. Bei mehreren möglichen Strassen kommt dann zum Beispiel folgende Dialogbox:

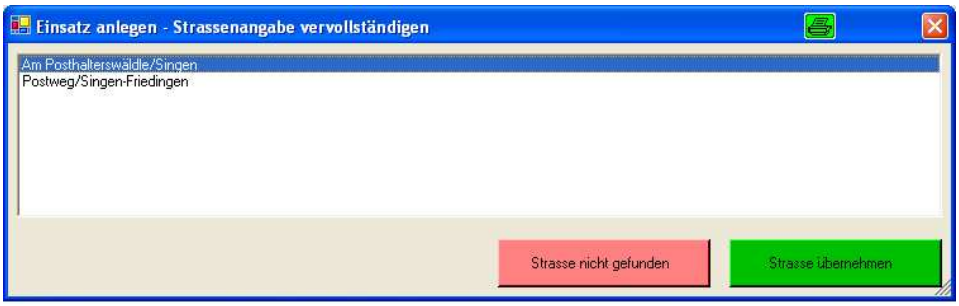

Abb. 4.3: Strassenauswahldialog bei mehreren Möglichkeiten

Dieses Beispiel soll dazu dienen, die generelle Struktur des Systems EASy zu verdeutlichen.
Neben der eben gezeigten Kommunikationsform (Kommando mit erwarteter Antwort) sind noch zwei weitere Formen implementiert: Kommando ohne erwartete Antwort und Informationsnachrichten des Servers.
Die erste Form dieser Nachrichten wird im folgenden Beispiel dargestellt:

Quellcode 4.5: Beispiel für eine XML-Nachricht - FMS-Signal

```
<EASy commPartnerName="FMSAuswerter1Queue">
  <EASyCommandname="FMSSignal">
    <dataFMS xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <FMSKennung>62666345</FMSKennung>
      <DatumUhrzeit>10.08.2003 21:35:27</DatumUhrzeit>
      <Status>2</Status>
      <Tki>2</Tki>
      <Richtung>0</Richtung>
      <Baugruppe>1</Baugruppe>
      <Folgetelegramm />
      <quittiert>true</quittiert>
      <FahrzeugID>0</FahrzeugID>
      <Quelle />
      <EinsatzID>0</EinsatzID>
    </dataFMS>
  </EASyCommand>
</EASy>
```

Hier kommt ein FMS-Signal vom Fahrzeug mit der Kennung 62666345, das den Status 2 setzt. Diese Nachricht wird vom Server nun verarbeitet. Zu dieser Verarbeitung gehört neben der Speicherung in der Datenbank, der Zuordnung zu einem eventuell laufenden Einatz und anderen Verwaltungstätigkeiten auch, dass eine entsprechende Information über den Statuswechsel an die angeschlossenen EASyWindowsClients gesendet wird. Bei diesen Nachrichten handelt es sich um die zweite der angesprochenen zusätzlichen Formen.
Diese sieht nun folgendermaßen aus:

Quellcode 4.6: Beispiel für eine XML-Inform-Nachricht - FMS-Signal

```
<?xml version="1.0" encoding="utf-8"?>
<EASy>
  <EASyInform name="FMSSignal">
    <dataFMS xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <FMSKennung>62666345</FMSKennung>
      <DatumUhrzeit>15.08.2003 21:35:27</DatumUhrzeit>
      <Status>2</Status>
      <Tki>2</Tki>
      <Richtung>0</Richtung>
      <Baugruppe>1</Baugruppe>
      <Folgetelegramm />
      <quittiert>true</quittiert>
      <FahrzeugID>137</FahrzeugID>
      <Quelle />
      <EinsatzID>0</EinsatzID>
    </dataFMS>
  </EASyInform>
</EASy>
```

Die angesprochenen Clients erhalten diese Nachricht in den jeweiligen Client-Queues, im Beispiel von EASyWindowsClient1 heisst die entsprechende Queue **EASyWindowsClient1**. Neben Informationsnachrichten über Statusänderungen, Änderungen in Einsätzen, etc. kommen auf diesem Weg auch Kommandos an die Clients, die diese veranlassen, Stammdaten neu einzulesen, Archivfunktionen durchzuführen, etc.
Für Wartungsarbeiten arbeitet im EASy-Server ein Timer, der im Minutentakt die Datenbank auf anstehende Wartungsarbeiten überprüft und diese dann entsprechend ausführt. Dabei muss der Adressat der Aufgabe nicht unbedingt der Server sein, auch ein Client kann entsprechende Steueranweisungen erhalten.
Ebenfalls per Message-Queueing ist die Synchronisation mit anderen Servern geplant, die Ihrerseits wieder Ihre Clients mit Daten und Kommandos versor-

gen. Diese Funktionalität ist zum Ende der Diplomarbeit allerdings noch nicht implementiert, soll jedoch Einzug in ein späteres Release halten.
Die gewählte Struktur ermöglicht den Betrieb vieler Clients gleichzeitig und hält diese synchron, ohne dass diese ein aktives Polling[14] betreiben müssen. Dadurch entsteht ein Performance-Vorteil. Desweiteren findet eine Kontrolle der eingegebenen Daten auf dem Server auf Plausibilität und Konsistenz statt, es kann somit nicht zu Überschneidungen kommen. Auch dies ist ein klarer Vorteil der hier gewählten Struktur.
Als nachteilig erweist sich allerdings der große Aufwand bei der Entwicklung, da für jede Funktion insgesamt 5 oder mehr Methoden entwickelt werden müssen:

- Die Funktion auf Client-Seite inklusive Übertragungsmethode (Sende-Richtung)
- die Zuordnung der Nachricht im EASy-Server
- die Verarbeitung der Nachricht im Server
  - die Erstellung einer Antwort im Server oder
  - die Erstellung einer beziehungsweise mehrerer EASyInform-Nachrichten für die Clients
- die Verarbeitung der Antwort beziehungsweise der EASyInform-Nachricht im Client

Ein Ansatz, der ein direktes Schreiben in die Datenbank vorsieht kommt ohne diesen zusätzlichen Aufwand aus, vereint allerdings auch die oben erwähnten Vorteile als Nachteil in sich. Vor allem ist hier die deutlich höhere Netzwerklast zu nennen, da von Client-Seite ständig bei der Datenbank nachgefragt werden muss, ob Veränderungen vorliegen.

[14] aktive Abfrage zum Beispiel einer Datenbank auf neue Daten, basierend zum Beispiel auf einem Zeitstempel

4.2.2 Klassenentwurf

Beim Entwurf der Klassen für EASy habe ich darauf geachtet, eine möglichst große Wiederverwendbarkeit einmal erstellter Software-Teile zu erreichen und bei Änderungen an den Datenbankstrukturen (zum Beispiel durch das Hinzufügen zusätzlicher Datenfelder) einen möglichst geringen Aufwand zu haben.
Um diese Ziele zu erreichen habe ich die zwei Bibliotheken **EASyBibliothek** und **EASyBibliothekUI** entwickelt, die Code enthalten, der in allen Teilen des EASy-Systems verwendet wird. In der ersten Bibliothek sind vor allem Strukturklassen enthalten, die von allen Programmteilen verwendet werden. Daher muss eine Änderung zum Beispiel eines Feldtyps oder das Hinzufügen eines Feldes lediglich in dieser Strukturklasse und in den verwendenden Klassen durchgeführt werden, nicht jedoch in den Methoden, die für die Datenübertragung zwischen dem Server und den anderen Bestandteilen des EASy-Systems zuständig sind, da hier eine Serialisierung und Deserialisierung eines Objekts des Typs der Strukturklasse erfolgt und keine fest codierten Strukturinformationen enthalten sind.
Diese Strukturklassen enthalten die Attribute und teilweise Zugriffsmethoden auf die Attribute.
Exemplarisch sei hier die Klasse **dataEinsatz** gezeigt, die momentan die komplexeste Klasse im EASy-System ist. Gut zu erkennen ist, dass in dieser Klasse etliche ArrayLists (also Listen, die Objekte einer bestimmten Klasse in beliebiger Länge enthalten) vorkommen. Sollte in der Zukunft nun Bedarf an der Speicherung weiterer Werte bestehen, ist es nur notwendig, die Strukturklasse und die verarbeitenden Funktionen entsprechend zu erweitern, die Kommunikation wird sich automatisch anpassen.

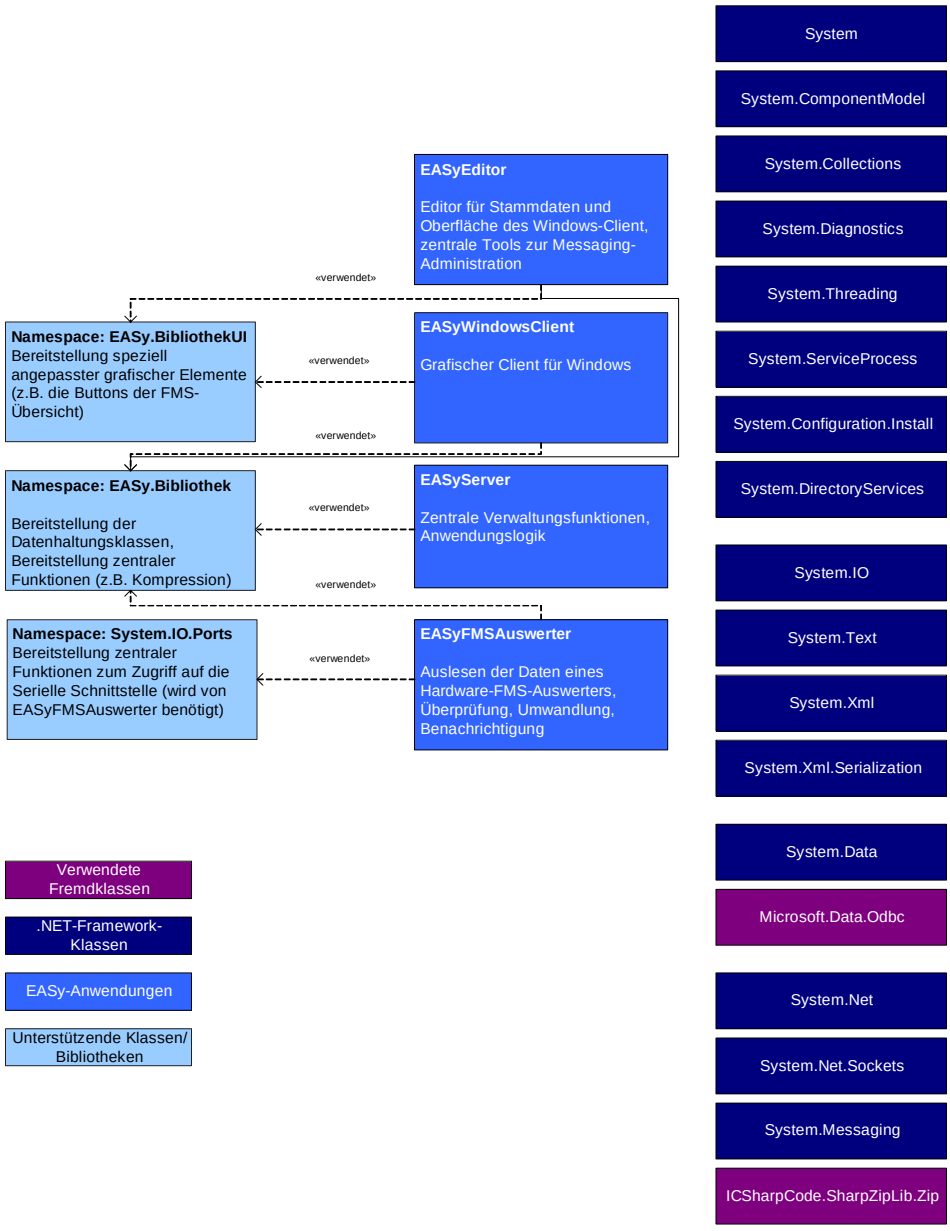

Abb. 4.4: Klassenstruktur von EASy

Quellcode 4.7: Die Strukturklasse dataEinsatz

```
public class dataEinsatz
{
    public string EinsatzID="0";
    public string StichwortID="";
    public dataEinsatzEinsatzstelle aktEinsatzstelle;
    public string ZustaendigerStandortID="";
    public dataAAO gueltigeAAO=new dataAAO();
    public int Prio=0;
    public string Kommentar="";

    [XmlArrayItem(typeof(dataEinsatzProtokoll))]
    public ArrayList EinsatzProtokoll=new ArrayList();

    [XmlArrayItem(typeof(dataEinsatzZeit))]
    public ArrayList EinsatzZeiten=new ArrayList();

    [XmlArrayItem(typeof(dataEinsatzBericht))]
    public ArrayList EinsatzBericht=new ArrayList();

    [XmlArrayItem(typeof(dataEinsatzMaterial))]
    public ArrayList EinsatzMaterial=new ArrayList();

    [XmlArrayItem(typeof(dataEinsatzPersonal))]
    public ArrayList EinsatzPersonal=new ArrayList();

    [XmlArrayItem(typeof(dataEinsatzSchleife))]
    public ArrayList EinsatzSchleifen=new ArrayList();

    public int EinsatzStatus=0;

    public string EinsatzBeginn="";
    public string EinsatzEnde="";



    public dataEinsatz()
    {
        aktEinsatzstelle=new dataEinsatzEinsatzstelle();
    }

```

```
        public void setZustaendigerStandortID(
42        string ZustaendigerStandortID_)
        {
44          ZustaendigerStandortID=ZustaendigerStandortID_;
        }
46
        /*
48      (weitere Zugriffsmethoden set/get)
        */
50


52      public override string ToString()
        {
54          return aktEinsatzstelle.getStrassenname()+" "+
                   aktEinsatzstelle.getHausnummer()+" ("+
56                 aktEinsatzstelle.getName()+
                   ", Prio: "+Prio.ToString()+")";
58      }
    }
60
    public class dataEinsatzSchleife
62  {
        public string Schleife="";
64      public string Uhrzeit="";

66      public override string ToString()
        {
68          return Schleife+" ("+Uhrzeit+")";
        }
70  }

72  public class dataEinsatzProtokoll
    {
74      public long EinsatzProtokollID=0;
        public long EinsatzID=0;
76      public long EinsatzProtokollIDintern=0;
        public int EinsatzProtokollIDinternVersion=0;
78      public string EinsatzProtokollTyp="";
        public string EinsatzProtokollText="";
80      public string EinsatzProtokollPartner1="";
        public string EinsatzProtokollPartner2="";
82      public System.DateTime EinsatzProtokollZeit=
```

```
            new System.DateTime(1,1,1,0,0,0);
84      public int EinsatzProtokollRichtung=-1;
            // 0 Von aussen zum EASy-Bediener,
86          // 1 EASy-Bediener nach aussen

88      /*
        (Zugriffsmethoden set/get)
90      */
    }
92
    public class dataEinsatzMeta
94  {
        public string EinsatzMetaName;
96      public string EinsatzMetaString;
    }
98
    public class dataEinsatzZeit
100 {
        public long FahrzeugID=0;
102     public long EinsatzID=0;
        public string Zeit_Status3="";
104     public bool Zeit_Status3_manuell=false;
        public string Zeit_Status4="";
106     public bool Zeit_Status4_manuell=false;
        public string Zeit_Status1="";
108     public bool Zeit_Status1_manuell=false;
        public string Zeit_Status2="";
110     public bool Zeit_Status2_manuell=false;
        public string Zeit_Uebergabe="";
112     public bool Zeit_Uebergabe_manuell=false;

114     public string Kilometer="";
        public string Betriebsstunden="";
116
        /*
118     (Zugriffsmethoden set/get)
        */
120 }

122 public class dataEinsatzBericht
    {
124     public long FeldID=0;
```

```
        public string Typ="";
126     public string Inhalt="";
        public bool Checked=false;
128
        /*
130     (Zugriffsmethoden set/get)
        */
132 }

134 public class dataEinsatzMaterial
    {
136     public long EinsatzMaterialID=0;
        public long EinsatzID=0;
138     public long MaterialID=0;
        public long FahrzeugID=0;
140     public float Menge=0;
        public string Bemerkung="";
142
        /*
144     (Zugriffsmethoden set/get)
        */
146 }

148 public class dataEinsatzPersonal
    {
150     public long EinsatzPersonalID=0;
        public long EinsatzID=0;
152     public long PersonalID=0;
        public long FahrzeugID=0;
154     public string FahrzeugPosition="";
        public bool PAgetragen=false;
156     public string Bemerkung="";

158     /*
        (Zugriffsmethoden set/get)
160     */
    }
162
    public class dataEinsatzEinsatzstelle
164 {
        public string StrassenID;
166     public string Strassenname;
```

```
        public string Hausnummer;
168     public string Name;
        public string Ort;
170
        /*
172     (Zugriffsmethoden set/get)
        */
174 }
}
```

Neben diesen Strukturklassen sind auch Funktionen wie zum Beispiel die Kompression von Strings in **EASyBibliothek** enthalten, die überall im System gebraucht werden.
Um eine spätere Portierung des Servers auf ein anderes Betriebssystem (siehe Abschnitt 4.1.4) zu erleichtern, wurden die speziell für EASy entwickelten graphischen Elemente in der Bibliothek **EASyBibliothekUI** ausgelagert. Damit ist es möglich, die Server-Bestandteile (der Server selbst, die Anbindung an externe Hardware, zum Beispiel FMS-Auswerter, etc.) zu portieren, auch wenn im .NET-kompatiblen Framework auf dem Zielsystem noch keine graphischen Oberflächenkomponenten implementiert sind.
Neben diesen selbst entwickelten Bibliotheken wurde noch auf die zusätzliche Bibliotheken **System.IO.Ports**(@Mi03b) zur Kommunikation mit externer Hardware über die serielle Schnittstelle und **SharpZipLib**(@IC03) zur Kompression von Strings zurückgegriffen.
Neben diesen eigen- beziehungsweise fremdentwickelten Bibliotheken kamen noch zahlreiche Bibliotheken des .NET-Frameworks zum Einsatz. Extensiv wurden vor allem die Klassen und Methoden der **System.Windows.Forms** und **System.Xml**-Namespaces genutzt.

4.2.3 Datenbankentwurf

Beim Datenbankentwurf habe ich darauf geachtet, nur Standard-SQL-Funktionen zu verwenden, um einen Austausch des zugrundeliegenden Relationalen Datenbanksystems problemlos zu gewährleisten, sollte der MySQL-Einsatz aus wie auch immer gearteten Gründen nicht mehr möglich sein. Die Datenbank wird in EASy als Speicherort für Stammdaten und Bewegungsdaten verwendet. Sämtliche Änderungen, zum Beispiel an einem Einsatzbericht werden nach Übertragung zum Server zuerst gespeichert, dann verarbeitet und anschließend erneut aus der Datenbank gelesen, bevor dieses Ergebnis im Rahmen einer EASyInform-Nachricht wieder an die Clients gelangt. So kann sichergestellt werden, dass die Daten im Hauptspeicher und in der Datenbank konsistent sind.

Die implementierte Datenbankstruktur ist in Abbildung 4.5 zu sehen. Die Tabellenbezeichner sind so gewählt, dass eine sofortige Erkennung der in der Tabelle gespeicherten Daten möglich ist, das heißt in den Tabellen, die mit

- **system_** beginnen, handelt es sich um Tabellen mit Systeminformationen,
- bei **einsatz_**-Tabellen um Tabellen, in denen Informationen und Bewegungsdaten der Einsätze gespeichert werden,
- bei den **fms_**-Tabellen handelt es sich um das FMS-Protokoll beziehungsweise -Archiv,
- ebenso wie das **zvei**-Protokoll.
- **fahrzeug**-Tabellen enthalten Daten zur Fahrzeugverwaltung,
- **stamm_**-Tabellen enthalten Stammdaten wie zum Beispiel Standorte, Personal, etc.,
- in den **aao_**-Tabellen sind Daten zur Alarm- und Ausrückeordnung hinterlegt und
- in den **adress_**-Tabellen sind schließlich die Adressen für die Adressdatenbank gespeichert.

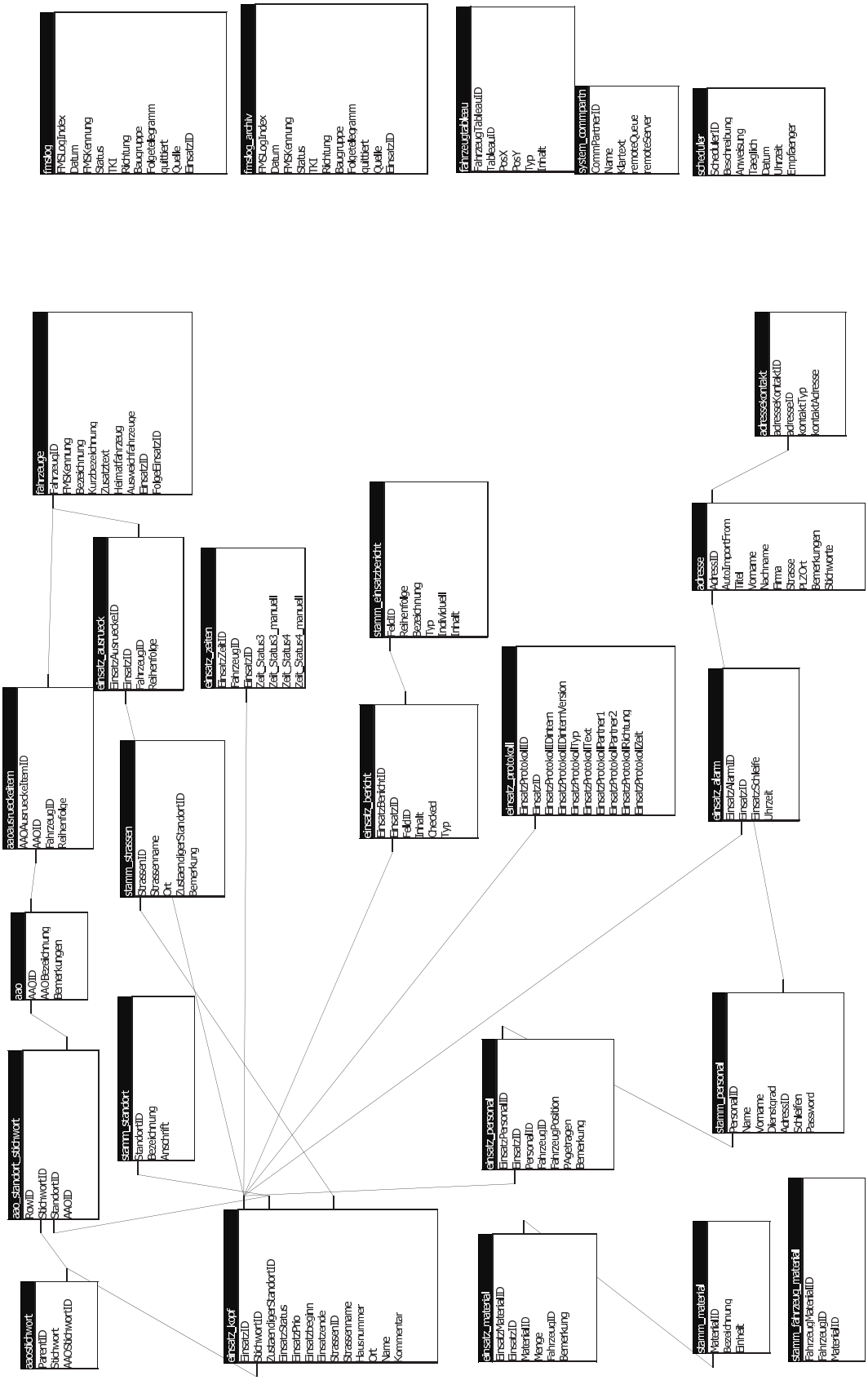

Abb. 4.5: DB-Struktur

4.3 Module

4.3.1 Servermodule

4.3.1.1 Kommunikation

Die Kommunikation der Clients mit dem Server findet wie bereits in Kapitel 4.2.1 erwähnt über ein Message Queueing-Verfahren statt. Im EASy-System kommen im aktuellen Versionsstand 65 verschiedene Nachrichtentypen vor. Dabei handelt es sich

- um einfache Strukturen wie das bereits mehrfach erwähnte FMS-Signal
- um komplexe Strukturen, wie zum Beispiel die Information über eine Änderung in einem Einsatzbericht oder die Statusänderung eines Fahrzeuges
- um Listen von Objekten (zum Beispiel um die Ergebnisse einer Adresssuche)
- oder um binäre Daten, wie zum Beispiel die Übermittlung eines Einsatzberichts im PDF-Format

Auf eine Auflistung und Ausführung der 65 Nachrichtentypen habe ich verzichtet, da diese Auflistung nur einen geringen Nutzwert für diese Diplomarbeit ergibt.
Über diese Kommunikationsschnittstelle ist auch die Anbindung externer Systeme realisiert, wovon ich weiter unten in diesem Kapitel berichten werde.

Wichtiges Element der Kommunikationsschnittstelle ist die eingebaute **Kompression**. Ziel dabei war es, die Daten die übermittelt werden so zu komprimieren, dass diese auch über langsame Leitungen (zum Beispiel eine Handy-Verbindung) übertragen werden können. Dazu wurde der ZIP-Algorithmus verwendet, um im Speicher die Nachrichten vor dem Versand zu komprimieren. Um eine Kompatibilität mit allen Clients zu erreichen wird im Server unterschieden, ob die Anfrage des Clients ebenfalls schon komprimiert war, bevor die Antwort komprimiert wird.

4.3.1.2 FMS-Verarbeitung

Im Server findet eine Verarbeitung der FMS-Signale statt. Dabei wird für die Verarbeitung nicht unterschieden, ob dieses Signal auf der Luftschnittstelle (also per FMS-Auswerter) empfangen wurde oder vom Anwender des EASy-Windows-Clients manuell eingegeben wurde. Diese Unterscheidung ist lediglich bei der Protokollierung der FMS-Aktivitäten relevant und wird auch dort gespeichert.

Als Besonderheit ist hier die automatische Übergabe eines Fahrzeugs in einen anderen Einsatz implementiert.
In der Praxis funktioniert dies folgendermaßen: Ein Feuerwehr-Fahrzeug ist in einem Einsatz eingebunden. Ein zweiter Einsatzauftrag kommt herein und muss bearbeitet werden. Das im Einsatz befindliche Fahrzeug kann sich (da zum Beispiel gerade aufgeräumt wird) aus dem Einsatzgeschehen des ersten Einsatzes herauslösen. Im Client wird daher für das Fahrzeug ein Folgeeinsatz angelegt. Das Fahrzeug sendet nun bei der Abfahrt von der ersten Einsatzstelle nicht das FMS-Signal 1 (Frei, unterwegs, erreichbar) sondern das FMS-Signal 3 (Einsatzauftrag übernommen). Damit wird das Fahrzeug automatisch aus der Verwaltung des ersten Einsatzes herausgenommen (natürlich gehen die bisher erfassten Daten nicht verloren) und dem Folgeeinsatz zugebucht. Damit findet zukünftig eine Zuordnung der FMS-Signale von diesem Fahrzeug zum Folgeeinsatz statt. Um diese Übergabe transparent zu gestalten, wird vom Server das Signal 3 in diesem Fall dupliziert, um den Zeitpunkt der Einsatzübernahme in beiden Einsätzen zu dokumentieren.

4.3.1.3 Einsatz-Verwaltung

Neben der Verwaltung der FMS-Signale werden die laufenden Einsätze vom EASy-Server verwaltet. Auch hier findet eine Überwachung der laufenden Einsätze statt, es ist angedacht, Timer anzusteuern, die zum Beispiel bei einer Überschreitung einer Frist automatisch Meldungen auf den anschlossenen EASy-Windows-Clients einblenden und auf die Situation aufmerksam machen.
Ebenso findet auf dem Server die Verarbeitung der Einsatzberichte und deren Weiterverarbeitung zum Beispiel für die Statistikfunktionen statt.

4.3.1.4 Datenspeicherung/Verwaltung

Der Server sorgt neben den bereits erwähnten Funktionen dafür, dass die zugrundeliegende Datenbank den gleichen Stand aufweist, der im Speicher gehalten wird. Ebenso werden vom Server regelmäßig Datenbankwartungsarbeiten veranlaßt, die dafür sorgen, dass die Performance des EASy-Systems gut bleibt.

4.3.1.5 Ausgabe-Funktionen

Um die Arbeitsstationen von mit großem Installationsaufwand verbundenen Aktivitäten wie zum Beispiel der PDF-Konvertierung der Einsatzberichte zu entlasten, habe ich diese Funktionen in den Server-Teil verlagert.
Am Beispiel des Einsatzberichtes möchte ich zeigen, wie hier vorgegangen wird:

- **Generierung der Einsatzdaten als XML-Dokument**: Im Server wird die interne Datenstruktur eines Einsatzes mit allen Aspekten in einem

XML-Dokument zusammengefasst. Dazu werden dann, ebenfalls als XML-Struktur, Stammdaten wie zum Beispiel die Personalliste, die Liste der Fahrzeuge und die Liste der Materialien der Fahrzeuge hinzugefügt. Um die Dimensionen zu verdeutlichen: Die komplette Struktur ergibt eine mehrere 100 KByte große Datei.

- **Umwandlung per XSLT**: Mittels eines Stylesheets (ca. 300 Zeilen lang) wird nun die im ersten Schritt erzeugte XML-Datei in ein TEX-Dokument umgewandelt.

- **Umwandlung in ein PDF-Dokument**: Mittels eines TEX-Interpreters und weiterer Tools wird nun das TEX-Dokument in ein PDF-Dokument umgewandelt.

- **Übertragung an den Client**: Der anfordernde Client erhält nun mittels einer XML-Nachricht in die das binäre PDF-Dokument eingebettet ist den entsprechenden Einsatzbericht und zeigt diesen auf dem Bildschirm an (s. Abb. 4.7). Anschließend kann er gedruckt oder zum Beispiel per E-Mail weiterversendet werden. Die dabei entstehende Datei hat eine durchschnittliche Größe von 30 KByte., ist also bestens für den Versand und die Archivierung geeignet.

1 Einsatzbericht Freiwillige Feuerwehr Singen (Nr: 137)

1.1 Kopfdaten

Einsatzstelle:	Bahnhofstrasse 3/Singen	Einsatzbeginn:	18.08.2003 17:13:00
	Martin	Einsatzende:	18.08.2003 18:20:35
Anforderung:	Leitstelle	Personalstärke:	3
Alarmierung:	FME	Fahrzeuge:	4
Einsatzart:	Kleinbrand B		
Menschen gerettet:			
Menschen tot:			

An der Einsatzstelle:			
✓	Rettungsdienst	✓	Oberbürgermeister
✓	Polizei		Ortpolizeibehörde
	Gas- und E-Werk		
✓	Kreisbrandmeister		
	Stadtwerke		
	Wirtschaftskontrolldienst		
	Wasserwirtschaftsamt		

Im Einsatz:			
✓	Abt. Kernwehr		Abt. Beuren a.d.A.
	Abt. Bohlingen		Abt. Friedingen
	Abt. Hausen a.d.A.		Abt. Schlatt u.K.
	Abt. Überlingen a.R.		

Einsatzschleifen:			
16301	17:13	16310	17:13

1.2 Fahrzeugübersicht

Fahrzeug	Ausgerückt	Ankunft	Rückfahrt	Eingerückt	Übergeben	km	Std.
Fl. Si 1/23-2 TLF 16/25	00.00. 00:00					0	
Fl. Si 1/33 DLK 23-12	00.00. 00:00					0	
Fl. Si 1/11 ELW 1	00.00. 00:00					0	
Fl. Si 1/45-2 LF 16-TS	18.08. 22:54	22:54	22:54	03.08. 22:54		5	

1.3 Personalübersicht

Name	Fahrzeug	Position	PA	Bemerkung
Tröndle,Stefan	Gerätehaus	Funk		
Egger,Andreas	1/11			
Martin,Andreas	1/45-2	ATF	✓	

1.4 Einsatzverlauf

gemeldete Lage bzw. Anforderung:	unklare Rauchentwicklung 1. OG
angetroffene Lage:	Kochtopf auf Herd stehen gelassen, Essen angebrannt, es gab Rindsgulasch
durchgeführte Massnahmen:	Wohnung durchgelüftet, Treppenhaus mittels Hochdrucklüfter belüftet, Wohnung an Bewohner und Polizei übergeben.
Sonstiges:	Bewohner: Hans Huber, Tel. 12345

1.5 Einsatzprotokoll

Zeit:	Partner 1		Partner 2	Meldung
18.08. 22:51	Zentrale 1; Tröndle, Stefan	←	ELW	Kochtopf auf Herd, keine Verstärkung
18.08. 22:51	Zentrale 1; Tröndle, Stefan	→	ELW	Wohnung leicht verraucht, wird belüftet
18.08. 22:52	Zentrale 1; Tröndle, Stefan	←	ELW	Einsatzkräfte kehren zurück

Abb. 4.6: Beispiel für die Druckausgabe eines Einsatzberichtes

1.6 Material

Bezeichnung	Einheit	1/23-2	1/33	1/11	1/45-2	Gesamt
C-Schlauch	Stk.				1	1

Abb. 4.7: Einsatzbericht, 2. Seite

4.3.2 Client-Module

4.3.2.1 EASy-Windows-Client

Neben dem Server gibt es noch mehrere Module, die zur Interaktion mit dem Anwender dienen. Wichtigstes Modul ist hier natürlich der EASy-Windows-Client als zentrales Anwenderprogramm. Der EASy-Windows-Client stellt alle Funktionen zur Verfügung, die für die normale Einsatzabwicklung notwendig sind. Nach einer kurzen Vorstellung der Oberfläche und des Bedienkonzeptes werde ich die wichtigsten dieser Funktionen vorstellen.

Oberflächengestaltung

Die Oberfläche des EASy-Windows-Client ist in folgende Abschnitte unterteilt:

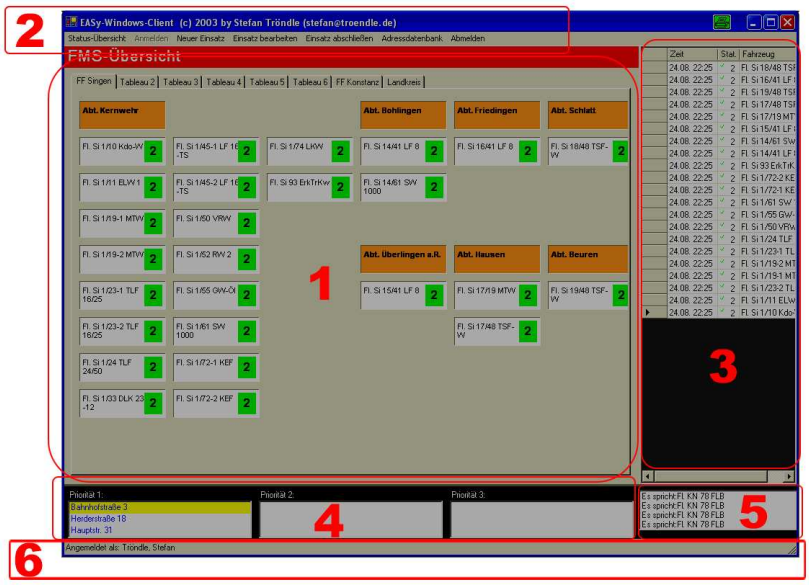

Abb. 4.8: EASy-Windows-Client: Oberflächenelemente

1. **Hauptfenster**: In diesem Fenster wird der wechselnde Inhalt (je nach gewählter Funktionalität) angezeigt.

2. **Menüleiste**: Hier werden die gewünschten Funktionen ausgewählt. In der Menüleiste sind die meisten Menüpunkte deaktiviert, wenn kein Benutzer angemeldet ist.

3. **FMS-Übersicht**: In der FMS-Übersicht werden die letzten FMS-Signale in umgekehrter chronologischer Reihenfolge angezeigt. Standardmäßig werden hier nur die FMS-Signale von Heimatfahrzeugen angezeigt, per Kontextmenü können jedoch alle Fahrzeuge eingeblendet werden.

4. **Einsatzübersicht**: In der Einsatzübersicht werden alle aktiven Einsätze aufgelistet. Die Farbe des Eintrags-Hintergrundes gibt an, welchen Fortschritt der Einsatz genommen hat, ob also schon ein Fahrzeug zu diesem Einsatz ausgerückt ist, ob der Einsatz abgeschlossen werden kann, etc.

5. **„Es spricht"-Fenster**: In diesem Fenster werden die betätigten Sprechtasten angezeigt, das heißt wenn ein Fahrzeug funkt (und es eine FMS-Ausstattung hat) wird hier der Name des Fahrzeugs angezeigt. Sinnvoll ist diese Funktion, wenn zum Beispiel aufgrund einer Störung im Funk der Name des rufenden Fahrzeugs nicht genau verstanden wurde.

6. **Statusleiste**: In der Statusleiste wird der aktuell angemeldete Benutzer angezeigt.

Tastaturbedienung

Der EASy-Windows-Client ist dafür ausgelegt, komplett mit der Tastatur bedient zu werden. Dazu wurden den meisten Funktionstasten (teilweise in mehreren Ebenen) Funktionsaufrufe zugewiesen. Gerade für Vielbenutzer ist dies ein zusätzlicher Komfortgewinn, da der häufige Wechsel zwischen Tastatur und Maus entfällt.

Funktionen

- **FMS-Übersicht**: In der Maske FMS-Übersicht wird, unterteilt in verschiedene Register, der aktuelle Zustand des Fuhrparks der entsprechenden Feuerwehr angezeigt. Um hier zum Beispiel bei Überlandhilfen[15] sich einen Eindruck vom dortigen Einsatzgeschehen zu machen beziehungsweise im Rahmen der Überlandhilfe angeforderte Fahrzeuge zu überblicken, können in den Registern alle erdenklichen Fahrzeuge und Beschriftungen gespeichert werden.

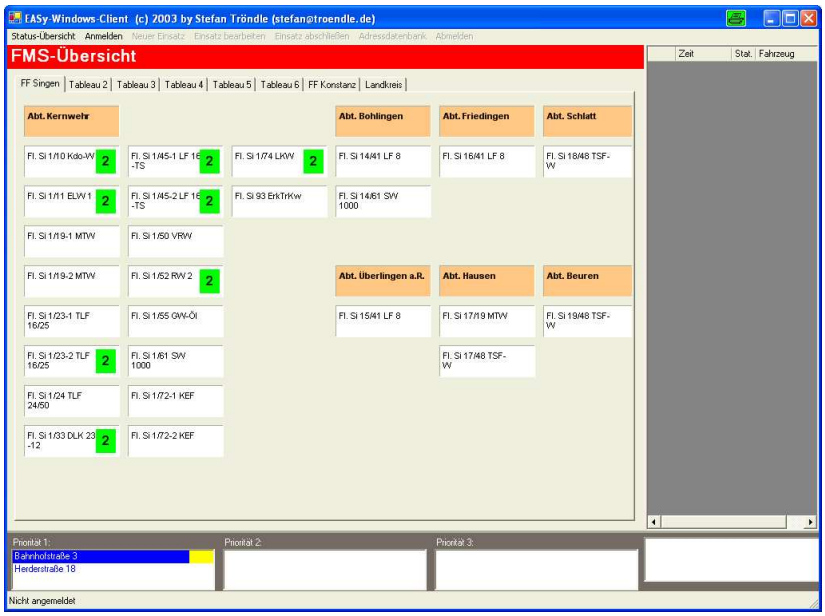

Abb. 4.9: EASy-Windows-Client: Maske „FMS-Übersicht“

[15] Feuerwehr ist Gemeindesache, das Tätigwerden zum Beispiel in der Nachbargemeinde nennt man Überlandhilfe

- **Anmeldung**: Um Veränderungen an den gespeicherten Daten vorzunehmen oder lesend darauf zuzugreifen, muss sich der Benutzer am EASy-System anmelden. Die Anmeldemaske wird automatisch aufgerufen, wenn der Benutzer zum Beispiel mit den Funktionstasten versucht, eine Aktion auszulösen. Die meisten Menüpunkte sind bis zur erfolgreichen Anmeldung gesperrt.

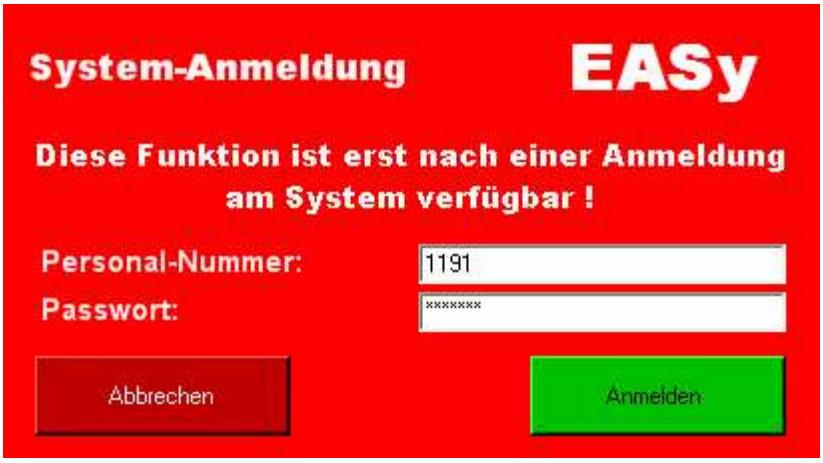

Abb. 4.10: EASy-Windows-Client: Systemanmeldung

- **Einsatz anlegen**:

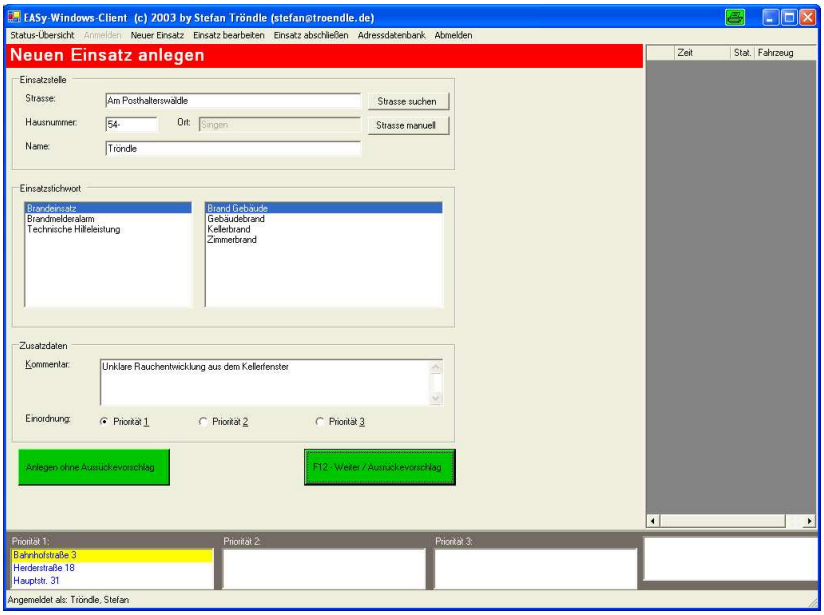

Abb. 4.11: EASy-Windows-Client: Maske „Einsatz anlegen“

In dieser Maske werden die wichtigsten Grunddaten eines Einsatzes erfasst, also die Adresse, und das Einsatzstichwort. Wenn die Adresse nicht eindeutig ist, wird eine Auswahlbox gezeigt, in der die passenden Straßen aus der Straßendatenbank angezeigt werden.

Abb. 4.12: EASy-Windows-Client: Maske „Straßenauswahl“

Nach der Auswahl der Straße und des Einsatzstichwortes wird vom EASy-Server die passendende Ausrückeordnung herausgesucht, die Fahrzeuge werden auf Verfügbarkeit überprüft und die Maske zur Bearbeitung der Ausrückeordnung wird dem Zentralisten angezeigt.
Wenn in dieser Maske nun Veränderungen vorgenommen werden, die nicht

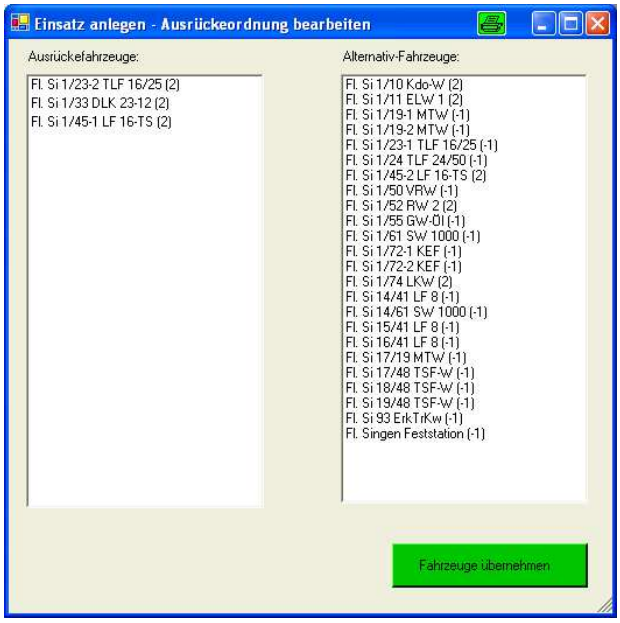

Abb. 4.13: EASy-Windows-Client: Maske „Einsatz anlegen: AAO bearbeiten“

vom Server selbst aufgelöst werden können, werden zusätzliche Entscheidungen vom Zentralisten angefordert:

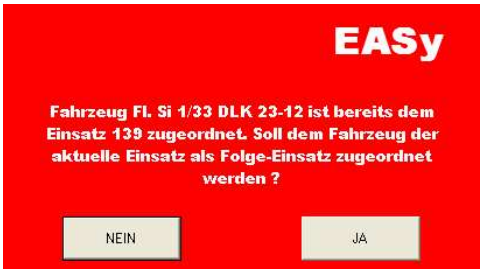

Abb. 4.14: EASy-Windows-Client: Maske „AAO-Entscheidungsabfrage“

Wenn alle Änderungen vorgenommen wurden, wird der Einsatz angelegt und steht nun zur weiteren Bearbeitung zur Verfügung.

- **FMS-Übersicht aktueller Einsatz**: Neben der oben erwähnten FMS-Übersicht über alle Fahrzeuge gibt es hier aus Gründen der besseren Übersichtlichkeit eine FMS-Übersicht der Fahrzeuge, die im aktuell gewählten Einsatz geführt werden.

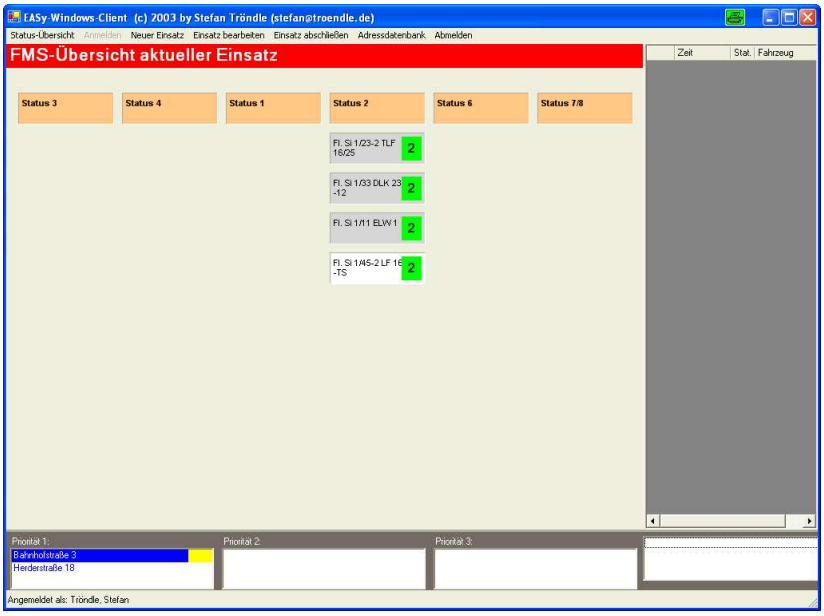

Abb. 4.15: EASy-Windows-Client: Maske „FMS-Übersicht aktuell“

Hier kann nun durch Rechtsklick auf das jeweilige Fahrzeug ein Kontext-Menü aufgerufen werden, mit dessen Hilfe dem Fahrzeug ein neuer Folgeeinsatz zugewiesen werden kann oder eine manuelle Statusänderung (zum Beispiel bei Ausfall des FMS-Systems) durchgeführt werden kann.

Abb. 4.16: EASy-Windows-Client: Maske „FMS-Übersicht-Kontextmenü“

- **Einsatzprotokoll**: Lagemeldungen und andere wichtige Informationen zum Einsatz werden im Einsatzprotokoll festgehalten.

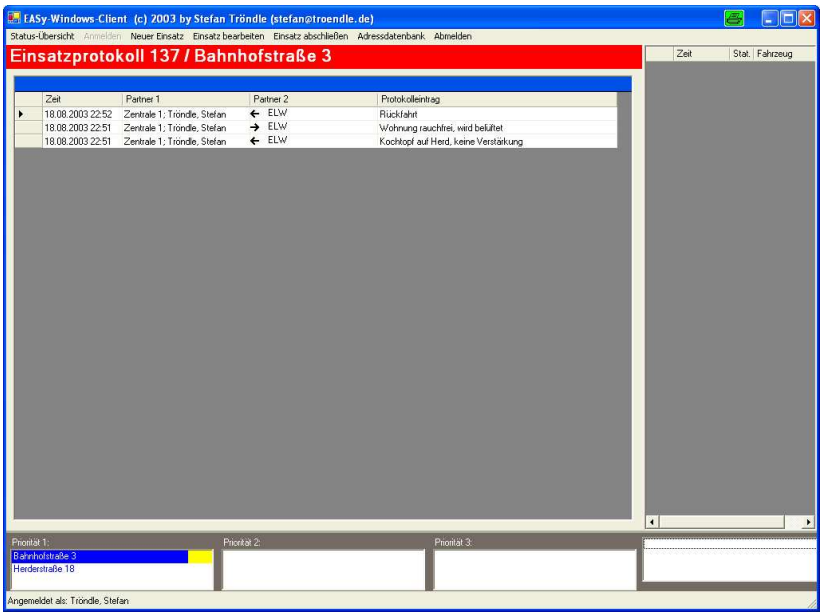

Abb. 4.17: EASy-Windows-Client: Maske „Einsatzprotokoll“

Neue Protokolleinträge werden über eine spezielle Eingabemaske erfasst:

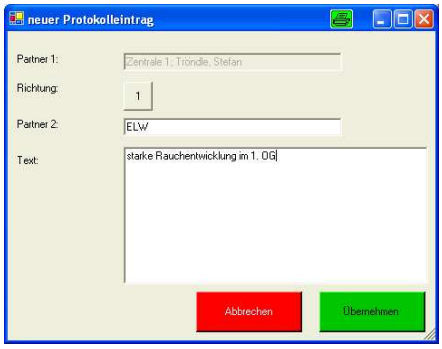

Abb. 4.18: EASy-Windows-Client: Maske „Protokolleintrag“

- **Einsatzbericht**: Im Einsatzbericht werden Daten wie zum Beispiel ein Bericht über die durchgeführten Tätigkeiten, andere anwesenden Hilfsorganisationen (Rettungsdienst, etc.) sowie die Art der Alarmierung erfasst. Die Maske ist nicht starr, sondern wird aus der Datenbank parametriert aufgebaut, damit hier eine Anpassung an die jeweiligen Bedürfnisse der anwendenden Feuerwehr möglich ist.

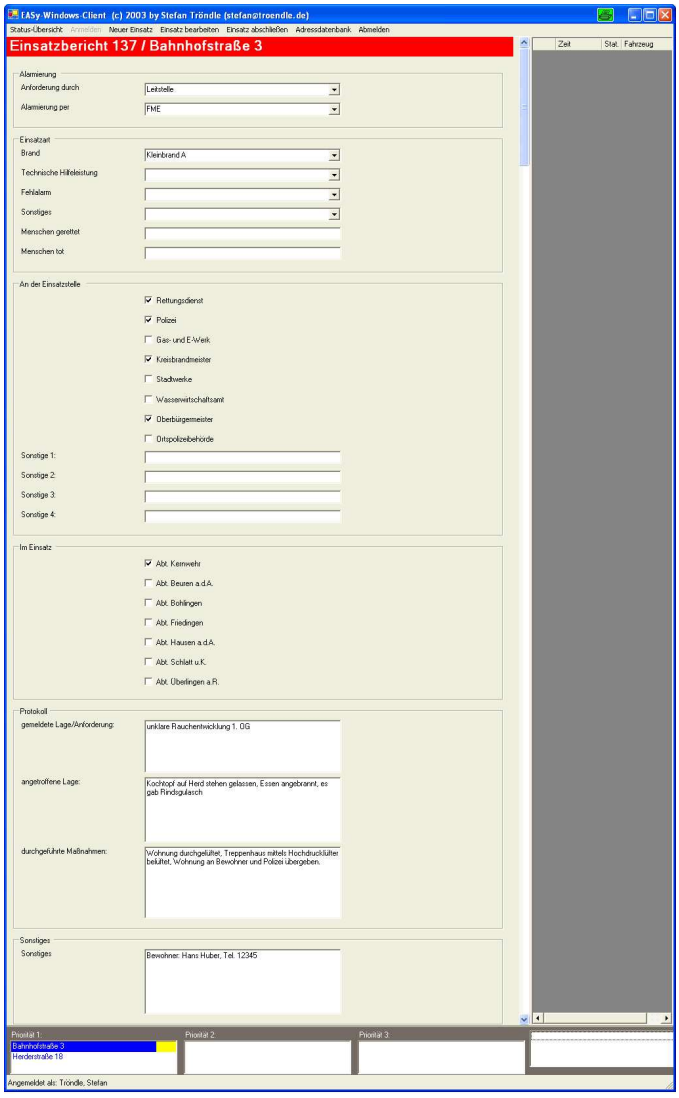

Abb. 4.19: EASy-Windows-Client: Maske „Einsatzbericht“

- **Einsatzbericht-Personal**: Hier wird über eine Eingabemaske das Personal, das im Einsatz tätig war mit allen dazugehörigen Daten erfaßt.

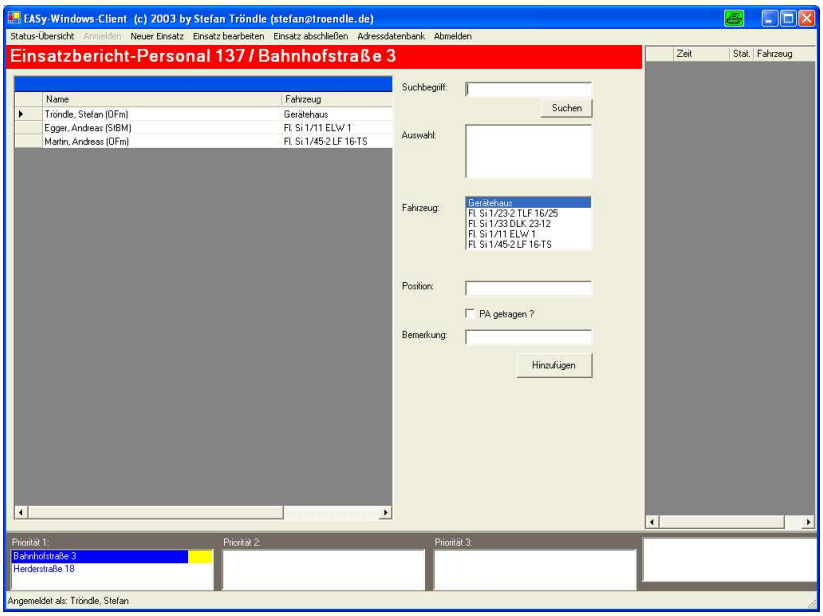

Abb. 4.20: EASy-Windows-Client: Maske „Einsatzbericht-Personal“

Die hier gewonnenen Daten werden einerseits für den Einsatzbericht verwendet, können aber später auch zu statistischen Zwecken verwendet werden. (zum Beispiel Berechnung der durchschnittlichen Einsatzbelastung des einzelnen Wehrmitgliedes)

- **Einsatzbericht-Material**: Hier wird das im Einsatz benötigte Material (die Anzahl der verwendeten Schläuche, Schaummittel das verbraucht wurde, etc.) erfaßt. Diese Maske wird ebenfalls aus Daten der Datenbank automatisch generiert. Die Materialien sind nur einmalig hinterlegt und mittels einer Zuordnungstabelle beim jeweiligen Fahrzeug hinterlegt, was später für den Einsatzbericht eine eindeutige Zusammenfassung der Mengen ermöglicht.

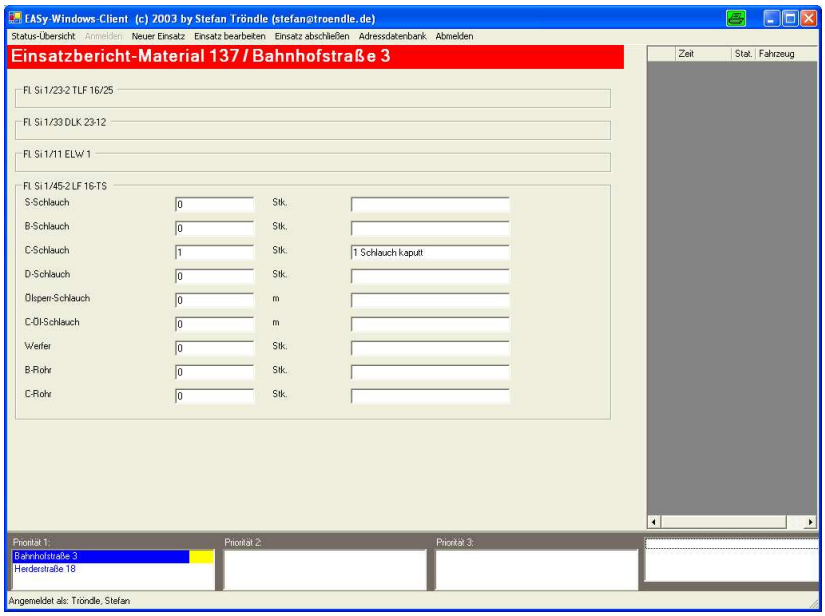

Abb. 4.21: EASy-Windows-Client: Maske „Einsatzbericht-Material“

- **Einsatz abschließen**: Nach Einsatzende werden hier die restlichen noch offenen Daten erfaßt. Darunter fallen auch die Ausrückezeiten der Fahrzeuge, die durch FMS-Signale bereits vorausgefüllt sind. Dem Zentralisten wird hier jeweils eine Liste der passenden Signale zur Auswahl angeboten, damit hier bei Falscheingaben (zum Beispiel im Fahrzeug wurde die falsche Taste gedrückt) noch nachträglich Korrekturen vorgenommen werden können.

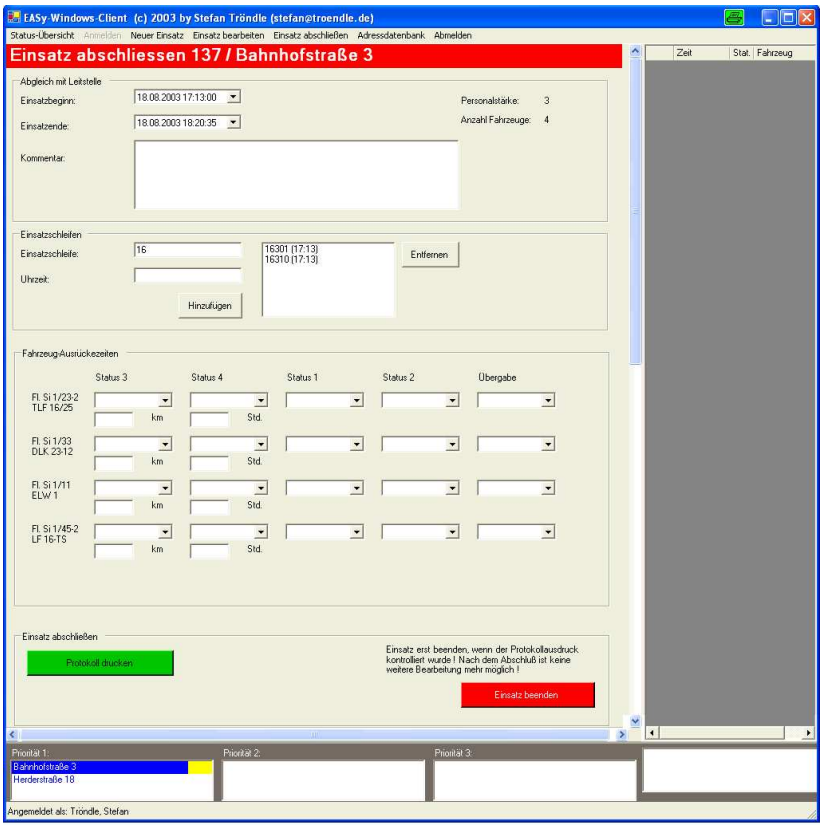

Abb. 4.22: EASy-Windows-Client: Maske „Einsatz abschließen“

Ebenfalls wird in dieser Maske der Druck des Einsatzberichtes ausgelöst. Ein Beispiel, wie der Einsatzbericht aussieht ist in Abbildung 4.7 zu finden.

- **Adressdatenbank**: Im System ist zusätzlich eine Adressdatenbank für die Kontaktinformationen der Wehrmitglieder und wichtiger Ansprechpartner der hinterlegten Objekte implementiert. Ebenfalls ist hier eine Stichwortsuche über zuvor hinterlegte Suchbegriffe möglich, um zum Beispiel schnell alle verfügbaren Betriebe aufzufinden, die einen Bagger zur Verfügung stellen können.

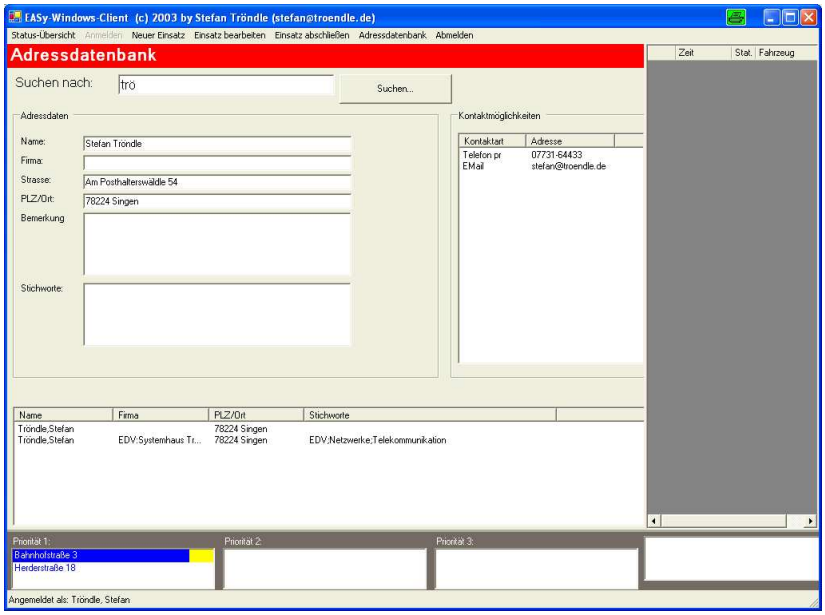

Abb. 4.23: EASy-Windows-Client: Maske „Adressdatenbank“

4.3.2.2 EASy-Editor

Der EASy-Editor ist ein weiterer wichtiger Bestandteil des fertigen EASy-Systems. Durch ihn wird es erst möglich, dass der „normale“ System-Administrator, also ein Administrator der nicht unbedingt Softwareentwickler ist, die Möglichkeit hat, Daten, Masken und Parameter des Systems zu pflegen.
Beim Editor kann durch den direkten Zugriff auf die Datenbank auf die Mehrfachimplementierung der einzelnen Funktionen verzichtet werden, desweiteren wird hier auf die Nutzung von Messaging verzichtet. Im Editor sind noch weitere Tools wie zum Beispiel der von mir entwickelte MSMQ-Explorer enthalten, der es ermöglicht, komplette Nachrichten aus einer Message Queue auszulesen[16]. Der Editor ist ein Softwareteil, der während des Praxisbetriebs sicher noch weiterentwickelt wird, um auch dem Administrator der jeweiligen EASy-Installation einen großen Bedienkomfort zu bieten.

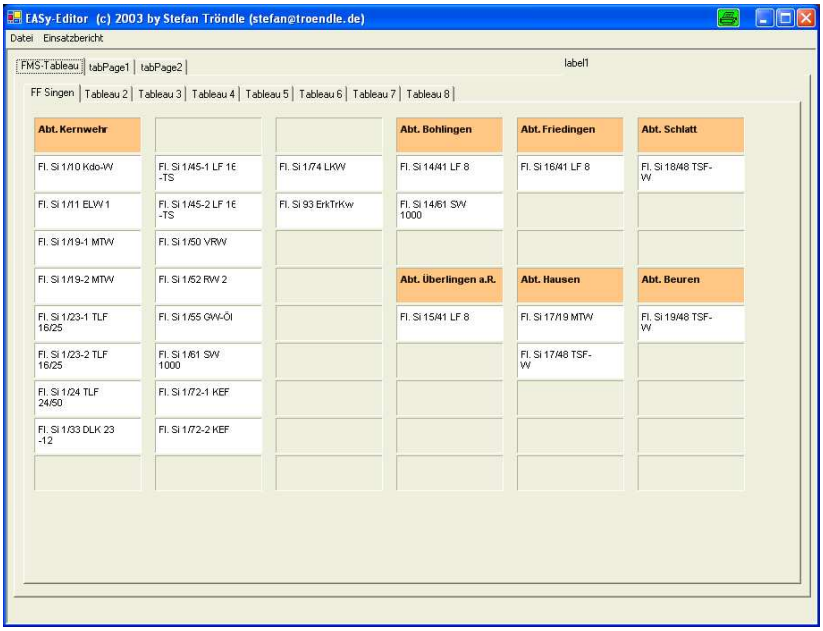

Abb. 4.24: EASy-Editor

[16]Mit Bordmitteln von Microsoft sind nur die ersten 270 Zeichen einer Nachricht auszulesen

4.3.3 Externe Schnittstellen/Erweiterungsmöglichkeiten

4.3.3.1 EASy-FMS-Auswerter

Ich möchte hier zuerst auf einige grundlegende Eigenschaften des FMS-Systems eingehen, bevor ich die Implementierung beschreibe:

Die FMS-Übertragung basiert auf dem Frequenzumtastverfahren (FSK[17]). Es wird dabei um eine Trägerfrequenz von 1500 Hz mit jeweils +/- 300 Hz getastet. Logisch „0“ entspricht einer Frequenz von 1800 Hz und logisch „1“ einer Frequenz von 1200 Hz. Der FMS-Leitrechner in der Leitstelle bestätigt das empfangene Signal und im Lautsprecher des Fahrzeuggeräts ertönt ein Aufmerksamkeitston mit einer Frequenz von 600 Hz. Die Übertragungsrate beträgt meist 1200 Baud. Das Telegramm wird auf dem selben Kanal übertragen, auf dem der normale Sprechfunkverkehr stattfindet.(@Fu03)

Übertragungsablauf eines FMS-Telegramms:

Block	Verwendungszweck	Bit-Länge	Zeit [ms]
	Sender-Vorlauf		200,0
	Telegramm-Vorlauf	12	10,0
	Block-Synchronisation	8	6,6
1	BOS-Kennung	4	3,3
2	Landes-Kennung	4	3,3
3-4	Orts-Kennung	8 (2 * 4 Bit)	6,6
5-8	Fahrzeug-Kennung	16 (4 * 4 Bit)	13,0
9	Status	4	3,3
10	B: Baustufenkennung	1	0,8
	R: Richtungsbit	1	0,8
	X, Y: TKI	2	1,6
11-12	Redundanz	7	5,8
	Stopbit	1	0,8
	Gesamt:	68	256,0

Auf dieses Signal folgt bei korrektem Empfang durch den Leitrechner der Leitstelle eine Quittung, die das empfangene Signal bestätigt. Aus dieser einfachen Regel resultiert leider auch das große Problem des FMS-Systems: Es gibt nur einen Leitrechner pro Funknetz, der sicher das gesendete Signal empfangen hat, nämlich den Leitrechner der das Signal quittiert hat, da es prinzipbedingt nur einen Quittungsgeber geben darf[18]. Da hier eine Sonderregelung gewaltige Kosten mit sich zieht, kann in der Folge mit einem vertretbaren Kostenaufwand nur

[17] Frequency Shift Keying

[18] Im Landkreis Konstanz gibt es eine Sonderkonstellation, die über softwaregesteuerte Filter realisiert wurde: Die Leitstelle in Radolfzell quittiert alle Signale außer denen, die von

passiv „mitgehört“ werden. Damit ist eben leider auch das Risiko verbunden, dass Signale aufgrund atmosphärischer Störungen nicht ankommen oder zum Beispiel eine Quittung nicht ankommt.
Da das passive „Mithören“ von FMS bei Feuerwehrleuten sehr beliebt ist, hat sich hier eine breite Masse von Software-Tools etabliert, die dieses ermöglichen. Ich habe mich bewußt aus Gründen der besseren Auswertung für den Einsatz eines Hardware-Auswerters entschieden. Dabei fiel die Wahl auf die FMS-Auswerter-Platine der Firma Barthfunk[19], mit der ich bei anderen Projekten schon gute Erfahrungen sammeln konnte.

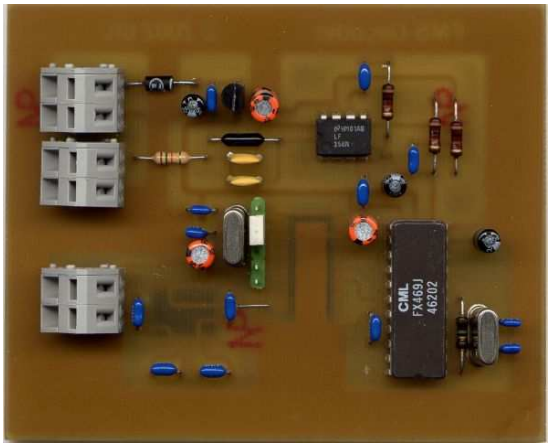

Abb. 4.25: Die Barthfunk-FMS-Auswerter-Platine

Diese Platine gibt ein empfangenes Signal auf der seriellen Schnittstelle in einem bereits dekodierten Format aus, so dass hier eine direkte Weiterverarbeitung in einer Software möglich ist. Der EASY-FMS-Auswerter liest nun die Daten von der seriellen Schnittstelle ein und wartet bis zu 250ms auf eine Quittung. Wenn diese Quittung erfolgt, wird das Signal entsprechend gekennzeichnet. Anschließend wird es an den Server gesendet (per Message Queueing). Weitere Überprüfungen finden im EASY-FMS-Auswerter nicht statt, eine Zuordnung ist rein im Server gegeben. Da dieser Software-Bestandteil recht einfach aufgebaut ist, ist er auch leicht an neue beziehungsweise andere Hardware anzupassen.

Konstanzer Fahrzeugen gesendet wurden. Diese werden direkt vom Einsatzleitrechner der Feuerwehr Konstanz quittiert.

[19] http://www.barthfunk.de

4.3.3.2 EASy-Personalerfassung

Als weitere Anbindung an ein externes System wurde eine elektronische Personalerfassung als Prototyp implementiert. Damit soll geprüft werden, ob diese Aufgabe statt wie bisher auf Papier auch praktikabel digital im Fahrzeug durchgeführt werden kann.
Als Hardware-Grundlage für das System habe ich C-Control[20] von Conrad Electronic ausgewählt. Dieses System ist ein mikrocontrollergesteuerter Mini-Computer, der verschiedene Schnittstellen zur Außenwelt anbietet. Mit einer Tastatur und

Abb. 4.26: C-Control Basis-Platine

einem LCD-Display wurde ein Gerät realisiert, das folgende Funktionalität erfüllt:

- Personaldatenerfassung für beliebig viele Einsätze (limitiert durch die Speichergröße des Geräts)
- Erfassung von beliebig vielen Personen pro Fahrzeug
- Erfassung, ob die Person Atemschutz (abgekürzt: PA = Pressluftatmer) getragen hat
- Übertragung der gespeicherten Daten nach Aufruf durch EASy.

[20] http://www.c-control.de

Im einzelnen sieht die Bedienung folgendermaßen aus:

- Nach Drücken der *-Taste startet das Programm.

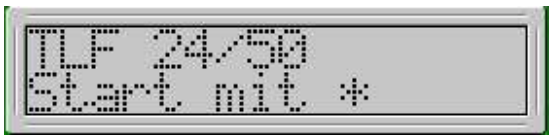

Abb. 4.27: Personaldatenerfassung, Bildschirm 1

- Nun muss die Einsatznummer eingegeben werden. Die Eingabe wird mit der *-Taste abgeschlossen, die #-Taste ermöglich einen Neubeginn der Eingabe

Abb. 4.28: Personaldatenerfassung, Bildschirm 2

- Es wird nun die erste Personalnummer eingegeben. Hier wird die Eingabe ebenfalls mit der *-Taste abgeschlossen, die #-Taste ermöglicht einen Neubeginn der Eingabe

Abb. 4.29: Personaldatenerfassung, Bildschirm 3

- Nach der Personalnummer wird erfaßt, ob die jeweilige Person ein Atemschutzgerät getragen hat. Weiter geht es nach Betätigen der 1 oder 0

Abb. 4.30: Personaldatenerfassung, Bildschirm 4

- Wenn keine weiteren Personen mehr auf dem Fahrzeug erfasst werden müssen, geht es mit der 0 zurück in den Startbildschirm.

Abb. 4.31: Personaldatenerfassung, Bildschirm 5

Die eingegebenen Daten werden im Meßwertspeicher des C-Control abgelegt und können bei Bedarf vom EASy-Server abgerufen werden.
Geplant ist, bei einem Realeinsatz des Gerätes eine Software zu realisieren, die im Rundrufverfahren die einzelnen C-Control-Geräte abfragt und bei Vorhandensein ausliest.
Ein Problem stellt noch die Anbindung der Geräte an den Computer dar. Hier kann aber eventuell auf die ebenfalls angebotene Funk-Telemetriedatenübermittlung zurückgegriffen werden.
Ich habe diesen sicherlich interessanten Ansatz verfolgt, um neben den Software-Komponenten von EASYauch eine Hardware-Komponente zu realisieren. Ob die Technik wirklich zum Einsatz kommt, hängt neben der Frage, wer die Geräte finanziert, auch sicherlich davon ab, wie die Anwender das Gerät akzeptieren.

4.3.3.3 EASy-CTI-Anbindung

Um das in EASy vorhandene Adressbuch ideal auszunutzen lag es natürlich auch nahe, eine Anbindung an die Telefonanlage mittels CTI (Computer Telephony Integration) zu realisieren.
Da die neue Telefonanlage für das Gerätehaus der Freiwilligen Feuerwehr Singen leider zum Ende der Bearbeitung der Diplomarbeit noch nicht installiert war,

konnte die geplante Funktionalität leider nicht getestet werden. Es ist jedoch angedacht, im EASy-Windows-Client eine Anbindung per serieller Schnittstelle an die jeweiligen System-Telefone zu realisieren, um den Wählvorgang gespeicherter Nummern direkt zu initiieren.

4.3.3.4 Anbindung eines GIS

In einem späteren Schritt ist die Anbindung eines GIS (Geo-Informations-Systems) geplant, um die Einsatzstelle/n auf einer Karte zu visualisieren. Gerade bei Flächenereignissen ist dies von großem Nutzen, um hier eine ideale Abfolge der Einsätze einzuplanen.
Geplant ist eine Anbindung an das System dEISy (Digitales Einsatzstellen-Informationen-System)[21], das ein Kommilitone im Rahmen seiner Diplomarbeit entwickeln wird.
Die Schnittstelle zu diesem System ist im Prinzip schon vorhanden, eine einfache Registrierung des GIS-Servers für EASy-Inform-Nachrichten reicht aus. Damit wird der GIS-Server automatisch über Änderungen des jeweiligen Einsatzstatus beziehungsweise über neue Einsätze informiert. Da sich hier die günstige Gelegenheit ergibt, mit anderen Studenten zusammenzuarbeiten wurde auf die Evaluierung weiterer Systeme für diese Funktion verzichtet, auch wenn dadurch die Funktionalität erst zu einem späteren Zeitpunkt realisiert werden kann.

[21] http://www.deisy.org

5 Vermarktung

EASy wurde nicht entwickelt, um in der Schublade zu verschwinden.

Obwohl viel Entwicklungsarbeit in EASy steckt ist mir natürlich auch bewußt, dass ein Softwaresystem eine so kritische Aufgabe wie die angestrebte nur dann erfüllen kann, wenn es ausreichend getestet ist. Wenn diese Tests abgeschlossen sind und der Funktionsumfang noch erweitert wurde, werde ich mich bemühen, das System am Markt zu platzieren.
Ich habe daher jetzt schon eine Marktübersicht erstellt, eine Zielgruppe für das System definiert und einen Marketing-Plan konzipiert. Das Ergebnis dieser Arbeit ist in den folgenden Abschnitten zu finden.

5.1 Marktübersicht

Auf dem Markt für Einsatzleitsysteme (von dem ich mich bewußt unterscheiden will) gibt es viele Systeme renommierter Hersteller. Leider sind nicht alle gewillt, Informationen auf Anfrage zu versenden, daher musste ich mich beim Vergleich größtenteils auf die Informationen verlassen, die ich im Internet finden konnte.
In den Vergleich einbezogen sind folgende Produkte, deren teilweise unterschiedliche Ausrichtung ich ebenfalls aufgeführt habe:

- **EASy**: Einsatz-Abwicklungs-System, soll unterstützende Tätigkeiten in der Abwicklung von Einsätzen bieten. Es ist nicht gewollt, Personalverwaltung, Materialverwaltung oder ähnliches zu betreiben, vielmehr ist EASy im jetzigen Entwicklungsstand auf ein anderes Programm angewiesen, das diese Aufgaben erfüllt.

- **secur.control**[1]: Einsatz-Leit-System von Wesser, ist im Einsatz bei der Integrierten Leitstelle des Landkreises Konstanz, dort konnte ich mir das System näher ansehen.

- **ELS pro**[2]: Deutsche Entwicklung, wird als Shareware vertrieben.

[1] http://www.wesser.com
[2] http://www.els-pro.de

- **ARIGON**[3]: Professionelles Feuerwehr-Verwaltungssystem, für das ebenfalls Module zur Einsatzabwicklung verfügbar sind.

In der folgenden Aufstellung habe ich versucht, die Systeme zu vergleichen. Ein ✓ steht dabei für eine vollständige Erfüllung der Funktion, ein † für die Möglichkeit der Erweiterung mit einem Zusatzmodul und ein leeres Feld für die Nicherfüllung einer Funktion.

Funktion	**EASy**	**secur.control**	**ELS pro**	**Arigon**
Netzwerkfähig	✓	✓	✓	✓
echtes Client-Server-System	✓	✓		
Verwaltungssystem (Personal/Material)				✓
Einsatztagebuch	✓	✓	✓	✓
Einsatzdisposition	✓	✓		
Alarmvorschlag		✓	✓	✓
Ausrückevorschlag (fahrzeuggenau)	✓	✓[a]	✓	✓
Fahrzeugalternativen	✓	✓[b]		✓
Zeitaufträge		✓	✓	
Fahrzeug-Status-Monitor	✓	✓	✓	✓
FMS-Anbindung	✓	✓	†	†
GIS-Anbindung	†[c]	†	†	†
Übergabe an FW-Verwaltungs-Software	†[d]	†		✓
Hinterlegung von Objekt-Informationen	†[e]	†	✓	†
Telefonlisten	✓	✓	✓	✓
Betriebssystem	Windows/ .NET-Plattformen	Windows/ Server: Unix	Windows	Windows

[a] Im Feuerwehrbereich nur eingeschränkt möglich
[b] manuell
[c] geplant
[d] geplant
[e] geplant

[3] http://www.swissphone.de

Natürlich gibt es Unterschiede im Detailgrad der jeweiligen Umsetzung, die ungefähre Ausrichtung der Systeme in diesem Marktsegment sollte jedoch klargeworden sein. Über die genauen Preisvorstellungen der einzelnen Anbieter war leider nicht viel zu erfahren, daher musste hier auf einen Vergleich verzichtet werden.

5.2 Zielgruppe

EASy soll, wie schon in der Einleitung angesprochen, die Zielgruppe der kleinen und mittleren Feuerwehren bedienen. Voraussetzungen für den erfolgreichen Einsatz von EASy sind:

- ein mittleres Einsatzaufkommen von 100-1000 Einsätzen pro Jahr, da ansonsten zu wenig Daten für sinnvolle Statistiken anfallen und der Zeitaufwand für Schulung, etc. zu groß ist.
- keine eigene Alarmabfrage sondern Alarmierung durch externe Stelle
- eine moderne EDV-Ausstattung
- eine beziehungsweise besser mehrere Verantwortliche, die die Datenpflege übernehmen.
- eine Feuerwehrverwaltungssoftware (zur Datenübernahme)
- verständnisvolle Zentralisten, die in der Lage sind, umzudenken
- verständnisvolle Trainer, die den Zentralisten das System und die Bedienung nahebringen.

5.3 Marketing-Plan

Nach einer erfolgreichen Einführung des Systems bei der Freiwilligen Feuerwehr Singen, Training der Zentralisten und Weiterentwicklung des Systems sind folgende Marketing-Maßnahmen denkbar:

- Erstellung einer EASy-Homepage
- Bewerbung der Homepage und des Produkts in Fachzeitschriften (Brandschutz, Brandhilfe, 112magazin, Feuerwehrmagazin)
- Bewerbung des Systems bei anderen Feuerwehren im Landkreis Konstanz, unter Benutzung vorhandener Kontakte

- Texten eines redaktionellen Beitrags für Fachpresse über das im Rahmen einer Diplomarbeit entstandene System
- Versand von Pressemitteilungen über die Markteinführung des Systems an die Fachpresse[4]
- Erstellen und Streuen eines Flyers auf Fachmessen und lokalen sowie überregionalen Veranstaltungen
- Veranstaltung von Präsentationsabenden, zum Beispiel mit Geschwindigkeitsvergleich zwischen manueller Bearbeitung und Unterstützung anhand eines Realbeispiels.
- Präsenz auf Fachmessen (zum Beispiel Rescue - Ausstellung und Fachkongress für interdisziplinäre Zusammenarbeit im Rettungswesen und in der Gefahrenabwehr, Stuttgart)
- Suche von Partnerfirmen, die EASy als Ergänzung zu eigenen Produkten, zum Beispiel Feuerwehrverwaltungssoftware anbieten
- Nutzen von persönlichen Kontakten zur Weiterverbreitung der Existenz von EASy, Nutzung des Referenzprojektes Feuerwehr Singen zur weiteren Kundengewinnung

Die Reihenfolge dieser Punkte stellt keine Wertung dar. Bevor das System allerdings vermarktet werden kann, muss es sich bei einem ausführlichen Test unter Realbedingungen bewähren und erweitert werden.

[4]Unterstützung durch Kats-Media Verein i.G., http://www.kats-media.org

6 Fazit

Natürlicher Verstand kann fast jeden Grad von Bildung ersetzen,
aber keine Bildung den natürlichen Verstand.

Arthur Schopenhauer, dt. Philosoph, 1788-1860

Übertragen auf EASy und ähnliche Systeme bedeutet dies, dass man sich nicht blind auf den Computer verlassen soll. Einerseits machen Entwickler Fehler beim Design und der Programmierung von Software, Administratoren machen Fehler bei der Datenpflege und Bediener tun auch nicht immer das, was sie eigentlich tun wollen. Damit nicht am Schluß der „blöde Computer“ Schuld ist, sollte jeder vor dem Klicken und Sprechen sein Gehirn einschalten und bemühen.

Auch ich habe versucht, während der Softwareentwicklung und dem Schreiben der vorliegenden Diplomarbeit mein Gehirn einzuschalten und nachzudenken, was ich programmiere oder schreibe. Trotzdem bleiben Fehler nicht aus, weder bei der einen noch bei der anderen Tätigkeit. Kleine Unsauberkeiten, die bei der besten wissenschaftlichen Arbeit vorkommen[1], bitte ich mir nachzusehen.

Was ich unbedingt noch loswerden möchte ist allerdings folgendes: **Es hat Spass gemacht !** Die Arbeit an dieser Diplomarbeit und der Software hat Spass gemacht, auch wenn es im Rekordsommer 2003 nicht immer einfach war, sich auf den Computer zu konzentrieren und die Aussentemperaturen zu vergessen. Doch mit viel Mineralwasser, geschlossenen Jalousien und einem Ventilator war diesen Problemen beizukommen. Schwieriger war es, manche Software-Fehler aufzuspüren, ob diese sich nun in der Eigenentwicklung, im .NET-Framework, in Büchern oder Anleitungen aus dem Internet befanden. Aber nachdem diese Hürden gemeistert waren (dem Internet sei Dank) ist doch ein vorzeigbares

[1] Man denke nur an den Absturz der ersten Ariane 5, der durch einen simplen Rechenfehler ausgelöst wurde.

Stück Software und Hardware (die gedruckte Form der Diplomarbeit ;-)) dabei herausgekommen, das einen praktischen Nutzen hat und im Laufe der nächsten Jahre einen noch größeren Nutzen bekommen wird.

Es war nicht einfach, eine gute Mischung zwischen Feuerwehr, Software und Wissenschaft zu finden, denn diese Dinge verstehen sich manchmal nicht ohne weiteres, doch ich freue mich, hier eine in meinen Augen gute Mischung gefunden zu haben.

Selbst wenn ich mich dazu entschließe, das Produkt nicht zu vermarkten war es nicht sinnlos, hier Arbeit und Gedanken zu investieren. Zwar erschrecke ich immer wieder, wenn ich meinen Arbeitsaufwand ansehe, denn der hat sich auf immerhin ca. 400 Stunden summiert. (Was das bei einem Stundenlohn von 80 EUR bedeutet, mag ich gar nicht ausrechnen...)
Der Gedanke, hier etwas eigenes geschaffen und im Rahmen des bei mir stark ausgeprägten Perfektionismus verfeinert zu haben, macht mich stolz und das ist mehr wert als viel Geld.

In diesem Sinne möchte ich mich vom Leser verabschieden.

Singen, im September 2003

Stefan Tröndle

Teil II
Anhang

Literaturverzeichnis

[@El03] @EldoS Corporation: *MsgConnect*
. http://www.msgconnect.com, 2003

[@Fu03] @Funkmeldesystem.de: *Aufbau des FMS-Datentelegramms der BOS*
. http://www.funkmeldesystem.de/bos-funk/fmsdat.html, 2003

[GHJV96] Gamma, Erich ; Helm, Richard ; Johnson, Ralph ; Vlissides, John: *Entwurfsmuster*. Bonn : Addison-Wesley, 1996. – ISBN 3–8273–1862–9

[Gün02] Günther, Karsten: *LATEX GE-PACKT*. Bonn : mitp-Verlag, 2002. – ISBN 3–8266–0785–6

[@Go03] @Golo Haas: *Was ist C#*
. http://www.golohaas.de/csharp, 2003

[Han02] Hanisch, Armin: *XML mit .NET - Programmierung und Basisklassen*. München : Addison-Wesley, 2002. – ISBN 3–8273–1998–6

[HS02] Herpers, Franz-Josef ; Sebestyen, Thomas J.: *XSL - Das Einsteigerseminar*. Landsberg : verlag moderne industrie Buch AG & Co. KG, 2002. – ISBN 3–8266–7209–7

[@IB03] @IBM Corporation: *Websphere MQ*
. http://www.ibm.com/software/integration/wmq, 2003

[@IC03] @IC#Code: *.NET Zip Library #ziplib (SharpZibLib)*
. http://www.icsharpcode.net/OpenSource/SharpZipLib/Default.aspx, 2003

[@Ja03] @Jabber, Inc.: *Jabber - Freedom to move ideas forward.*
. http://www.jabber.com, 2003

[Kof02] Kofler, Michael: *Windows Forms - Grafische Benutzerschnittstellen*. München : Addison-Wesley, 2002. – ISBN 3–8273–1994–3

[@La03] @Latz://new.media: *Blindtext-Archiv*
. http://www.newmediadesigner.de, 2003

[@Mi03a] @Microsoft Corporation: *GotDotNet: The Microsoft .NET Framework Community*
. http://www.gotdotnet.com, 2003

[@Mi03b] @Microsoft Corporation: *GotDotNet User Sample: SerialPort component and CSTerm terminal application*
. http://www.gotdotnet.com/Community/UserSamples/Details.aspx?SampleGuid=b06e30f9-1301-4cc6-ac14-dfe325097c69, 2003

[@Mi03c] @Microsoft Corporation: *Microsoft Developer Network*
. http://msdn.microsoft.com, 2003

[@Mi03d] @Microsoft Corporation: *Understanding Messaging with MSMQ*
. http://www.microsoft.com/windows2000/technologies/communications/msmq/msmqoverview.asp, 2003

[@My03a] @MySQL AB: *Datenbankserver MySQL*
. http://www.mysql.de/products/mysql/index.html, 2003

[@My03b] @MySQL AB: *MySQL-Benchmarks*
. http://www.mysql.de/information/benchmarks.html, 2003

[@Pr03] @PremiumSoft Cybertech Ltd.: *NAVICAT for MySQL*
. http://www.mysqlstudio.com, 2003

[@Sc91] @Schwarz, Norbert: *Einführung in TeX*
. http://www.ruhr-uni-bochum.de/www-rz/schwanbs/TeX/einfuehrung-in-tex.pdf : PDF-Ausgabe, 1991

[SR03] Schmengler, Andreas ; Ries, Uli: Netze ohne Grenzen. In: *PC Professionell* (2003), Nr. 9, S. 126–130

[VB02] Vaughn, William R. ; Blackburn, Peter: *ADO.NET - Examples and Best Practices for C# Programmers.* a!press, 2002. – ISBN 1–59059–012–0

Ehrenwörtliche Erklärung

Hiermit erkläre ich, Stefan Michael Tröndle, geboren am 23. September 1978 in Singen, ehrenwörtlich,

(1) daß ich meine Diplomarbeit mit dem Titel:
"Erstellung eines Einsatzabwicklungssystems für Feuerwehren“ an der Fachhochschule Konstanz im Fachbereich Informatik unter Anleitung von Professor Dr. Paul Wenzel selbständig und ohne fremde Hilfe angefertigt habe und keine anderen als in der Abhandlung angeführten Hilfen benutzt habe;

(2) daß ich die Übernahme wörtlicher Zitate aus der Literatur sowie die Verwendung der Gedanken anderer Autoren an den entsprechenden Stellen innerhalb der Arbeit gekennzeichnet habe.

Ich bin mir bewußt, daß eine falsche Erklärung rechtliche Folgen haben wird.

Konstanz, 18. September 2003 Stefan Tröndle